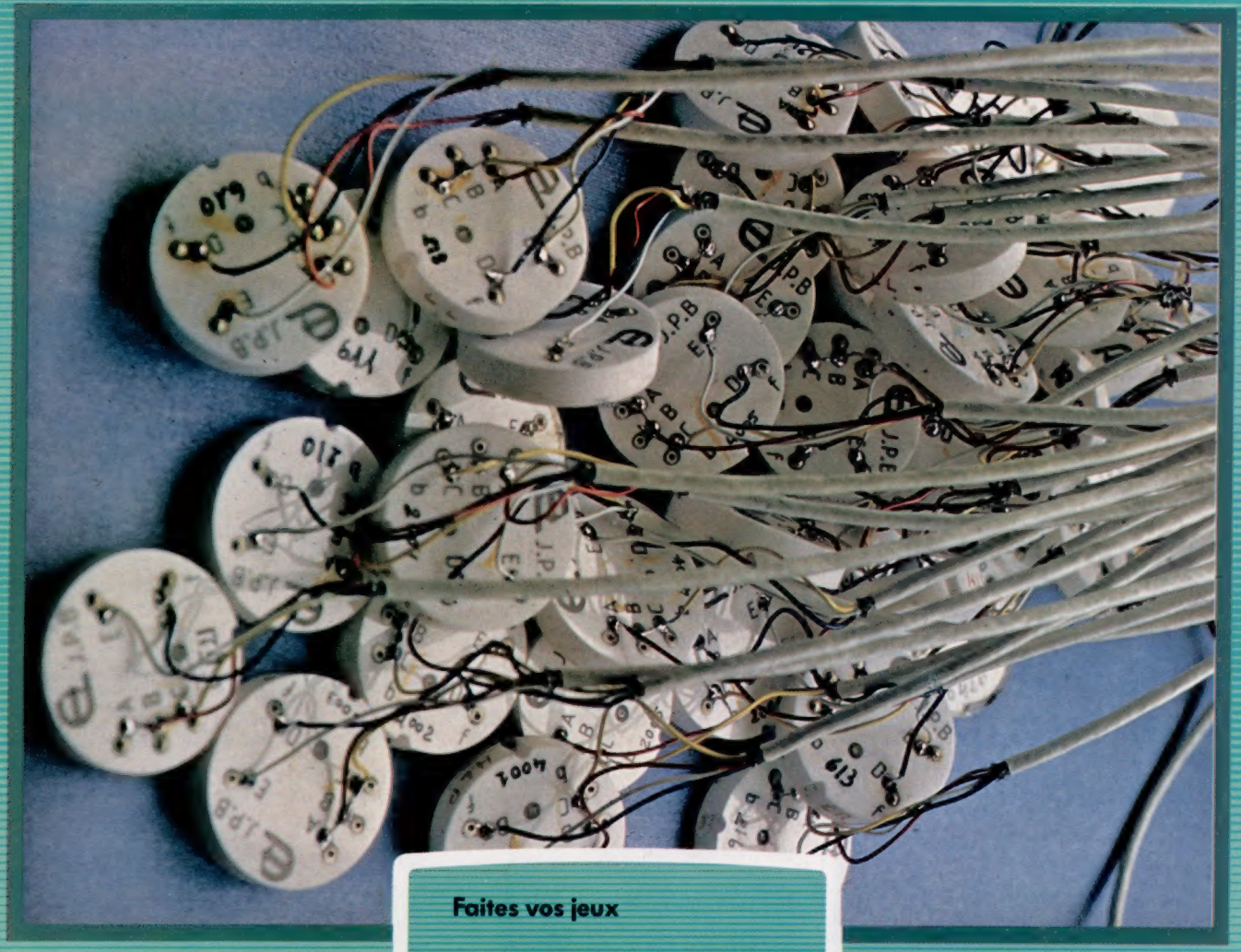
ABC INFORMATIQUE

N°27

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE



Faites vos jeux

Numération hexadécimale

Spectrum troisième version

La soudure et les micros

EDITIONS ATLAS

M6062-27-12F 85FB-3,80FS-$1.95

aBc INFORMATIQUE
cours d'informatique pratique et familiale
EDITIONS ATLAS

Édité par ÉDITIONS ATLAS s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : 538-52-70.

Belgique : ÉDITIONS ATLEN s.a., Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, Montréal Nord.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, Mezzovico.

Réalisé par EDENA s.a., 29, boulevard Edgar-Quinet, 75014 Paris. Tél. : 320-15-01.

Direction éditoriale : J.-Fr. Gautier. *Service technique et artistique* : F. Givone et J.-Cl. Bernar. *Iconographie* : J. Pierre. *Correction* : B. Noël.
Publicité : Anne Cayla. Tél. : 202-09-80.

VENTE

Les numéros parus peuvent être obtenus chez les marchands de journaux ou, à défaut, chez les éditeurs, au prix en vigueur au moment de la commande. Ils resteront en principe disponibles pendant six mois après la parution du dernier fascicule de la série. (Pour toute commande par lettre, joindre à votre courrier le règlement, majoré de 10 % de frais de port.)

Pour la France, s'adresser à ÉDITIONS ATLAS, tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : 538-52-70.

Pour les autres pays, s'adresser aux éditeurs indiqués ci-dessous.

SOUSCRIPTION

Les lecteurs désirant souscrire à l'ensemble de cet ouvrage peuvent s'adresser à :

France : DIFFUSION ATLAS, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., 55, avenue Huart-Hamoir, 1030 Bruxelles. Tél. : (02) 242-39-00. Banque Bruxelles-Lambert, compte nº 310-0018465-24 Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, 11450 boulevard Albert-Hudon, Montréal Nord, H 1G 3J9.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, zona industriale 6849 Mezzovico-Lugano. Tél. : (091) 95-27-44.

RELIEZ VOS FASCICULES

Des reliures mobiles permettant de relier 12 fascicules sont en vente chez votre marchand de journaux.

ATTENTION : ces reliures, présentées sans numérotation, sont valables indifféremment pour tous les volumes de votre collection. Vous les numéroterez vous-même à l'aide du décalque qui est fourni (avec les instructions nécessaires) dans chaque reliure.

En vente tous les vendredis. Volume III, nº 27.

ABC INFORMATIQUE est réalisé avec la collaboration de Trystan Mordrel *(secrétariat de rédaction)*, Jean-Pierre Bourcier *(coordination)*, Patrick Bazin, Jean-Paul Mourlon, Claire Rémy *(traduction)*, Ghislaine Goullier *(fabrication)*, Marie-Claire Jacquet *(iconographie)*, Patrick Boman *(correction)*.
Crédit photographique, couverture : Maillac-Rea.

Directeur de la publication : Paul Bernabeu. Imprimé en Italie par I.G.D.A., Officine Grafiche, Novara. Distribution en France : N.M.P.P. Tax. Dépôt légal : juillet 1984. 6847. Dépôt légal en Belgique : D/84/2783/27.

A NOS LECTEURS

En achetant chaque semaine votre fascicule chez le même marchand de journaux, vous serez certain d'être immédiatement servi, en nous facilitant la précision de la distribution. Nous vous en remercions d'avance.

Les Éditions Atlas

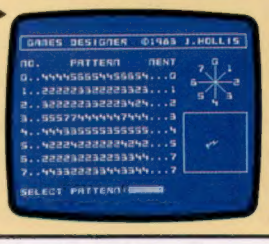
Faites vos jeux

On vend chaque année des millions de jeux pour micro-ordinateurs. Les firmes productrices de logiciels ont trouvé judicieux de proposer à leurs clients des programmes générateurs de jeux.

Créateur de jeux

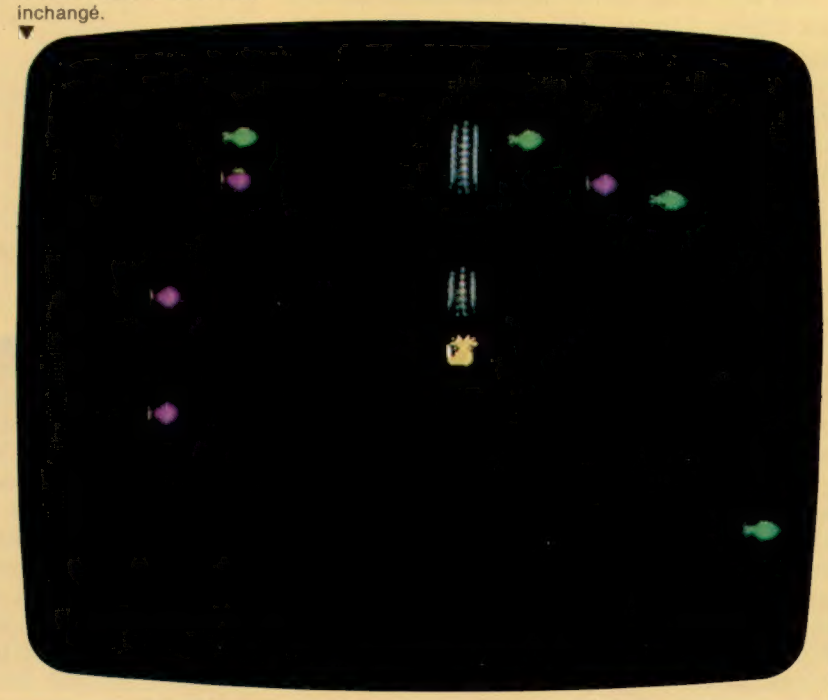
Configuration
Ce menu vous donne la possibilité de choisir la direction dans laquelle se déplace votre base, les couleurs du fond et de l'avant-plan, la façon dont apparaissent les extraterrestres, ainsi que les effets sonores.

Mouvement
Vous pouvez, grâce à ce menu, déterminer la manière dont surviennent les attaques. En haut et à droite de l'écran, un diagramme; en bas, un écran où viennent s'afficher les déplacements.



Au début
Les objets graphiques apparaissent d'abord sur un fond uni, et descendent de façon lente et ordonnée.

Résultat final
On voit ici quelles modifications ont été apportées aux éléments initiaux : seul le chat reste inchangé.



Création de figures
On peut remplir (ou effacer) chaque point en sélectionnant les coordonnées correspondantes. On peut voir en bas de l'écran la figure sous sa forme définitive (taille, couleur).

Ian McKinnell

Il existe deux grands types de « ludiciels » : les jeux d'aventures, auxquels le joueur doit participer activement, et qui peuvent ou non être illustrés graphiquement, comme c'est le cas de The Hobbit; et les jeux comme Space Invaders ou Asteroids, qui se limitent à l'écran même, au point que l'on pourrait y voir des « simulations imaginaires ».

Une analyse superficielle de ces deux genres montre les raisons de leur similitude. Pour prendre l'exemple de Space Invaders, il faut, pour jouer, deux éléments principaux : une base (vaisseau galactique, station spatiale, etc.) dotée d'une puissance de feu, et des cibles sur lesquelles on peut tirer. Il est donc possible, en modifiant à chaque fois ces deux paramètres de départ, d'obtenir une grande variété de jeux, différant entre eux par certains détails : position du défenseur, armes à sa disposition, rythme et intensité des vagues d'assaut. Il suffit d'ailleurs de comparer entre eux bien des programmes commerciaux d'allure semblable pour voir que c'est précisément ce que font les firmes productrices.

Un bon exemple de logiciel vous permettant de faire de même est le Games Designer (« Créateur de jeux ») de John Hollis — qui n'avait que seize ans à l'époque où il l'écrivit... Mis en vente par Software Studios/Quicksilva, il fonctionne sur Spectrum 48 K. Ce programme vous propose huit types de jeux fondamentaux. Une fois que vous en avez choisi un, il vous est possible d'en modifier les paramètres. Mais vous ne pourrez pas créer de jeux nouveaux; vous

devrez vous contenter de ceux qui vous sont présentés. Ils sont toutefois de bonne qualité.

La mise au point se fait par l'intermédiaire de menus. Une fois que vous avez fait votre choix, vous pouvez procéder à diverses modifications, en commençant par la forme et la couleur des objets graphiques qui représenteront les protagonistes. Ils sont déjà définis, mais c'est plus un avantage qu'un inconvénient dans la mesure où vous pouvez les modifier, bit par bit, au sein d'une matrice de dimensions 12 × 12 (c'est-à-dire un carré formé de quatre caractères). Vous pouvez également, si vous le désirez, partir d'une « feuille blanche » de même format. Les objets graphiques ou figures ainsi obtenus sont parfaitement lisibles et reconnaissables.

Le dessin de ces objets est lui-même guidé par un menu, qui affiche sur la gauche de l'écran les différentes options disponibles, et à droite une image agrandie du motif. Celui-ci est d'ailleurs reproduit en bas de l'écran, en dimensions et en couleurs « réelles », ce qui se révèle très utile.

On doit alors définir le mouvement de tous ces objets. C'est le rôle d'une partie du programme intitulée « Configuration ». Elle ne vous permet pas seulement le contrôle des directions de chaque protagoniste : à chaque fois il vous est proposé plusieurs types de mouvement que vous pouvez, si besoin est, fondre en un seul. Il vous faut ensuite décider des déplacements des éléments du décor.

Après quoi reste à régler la fréquence des vagues d'assaut, et à insérer certains effets spéciaux, visuels et sonores.

Une fois qu'on a créé un jeu, il convient de le sauvegarder sur cassette ou sur disquette, afin de pouvoir y jouer plus tard. C'est là que — du point de vue de l'utilisateur — Games Designer se révèle d'une grande insuffisance : il n'est pas possible de stocker le programme tel quel sur une mémoire auxiliaire. On ne peut préserver de cette façon que les données relatives aux paramètres qu'on a définis; pour jouer, il faut charger en mémoire Games Designer, puis les données elles-mêmes. Il est donc impossible de mettre au point un jeu particulier avec l'idée de le commercialiser plus tard...

The Quill

The Quill (« La Plume d'oie ») est, au contraire, expressément destiné à ceux qui désirent créer des jeux d'aventures et les mettre sur le marché. Il y a même dans le manuel d'accompagnement (il est très complet et fait plus de cinquante pages) une section intitulée « Comment vendre votre aventure ». Elle donne de précieux conseils pour tester son programme et s'assurer qu'il ne comporte pas d'erreurs. Le plus utile : faire jouer le plus de gens possible et écouter leurs avis. Les créateurs de The Quill ne demandent pas de droits d'auteur, simplement une mention précisant que vous avez utilisé leur programme. C'est une manifestation d'altruisme assez rare dans ce monde qui veut des vedettes. Il s'agit de créer un jeu du type « Donjons et dragons »; pourtant The Quill utilise des techniques assez proches de celles qui sont employées pour les bases de données nécessaires au traitement des problèmes de gestion. On distingue trois parties fondamentales : la base de données elle-même, un éditeur (qui permet la mise en œuvre des paramètres) et un interpréteur, qui permet une exécution interactive du jeu.

Il est d'usage de fixer une limite au nombre d'objets que le joueur peut transporter d'un endroit à l'autre. Le jeu est bien plus intéressant : il faut parfois aller et venir avant de pouvoir disposer de tout ce dont on aura besoin pour accomplir les tâches qu'impose le déroulement de l'intrigue; l'ingéniosité du programmeur peut se donner libre cours.

Choix d'objet
The Quill permet aux passionnés de créer leurs propres jeux d'aventures. Ce programme est assez semblable à certains logiciels de gestion ayant à traiter des bases de données. Il faut d'abord définir les lieux et les objets nécessaires au scénario. Des programmes objets permettent alors l'appel et l'affichage de ces attributs, ou leur emploi comme paramètres, chaque fois que le déroulement de l'intrigue nécessite leur apparition.
(Cl. Ian McKinnell.)

Pour mettre au point un jeu d'aventures, il faut d'abord rédiger un synopsis, élaborer un scénario et parfois dessiner une carte des lieux. Le manuel qui accompagne The Quill commence par présenter un jeu très simple, dans lequel le joueur a le choix entre dix objets répartis dans six pièces différentes. Astucieusement, ce jeu n'est pas intégré au programme lui-même : il vous faudra taper au clavier tous les paramètres, comme vous devrez le faire en créant votre propre histoire.

Le nombre de paramètres utilisables est naturellement limité, mais il est si grand (252 lieux et 210 objets) que vous ne serez sans doute jamais à court de possibilités.

Tournant sur Spectrum et sur Commodore 64, The Quill permet la création de jeux d'aventures qui ne le cèdent en rien à ceux qu'on trouve dans le commerce — et quelques-uns de ces derniers ont d'ailleurs été composés grâce à lui, comme ce jeu qui met en scène le mari de Mme Thatcher... Certaines restrictions n'en sont que plus regrettables. C'est ainsi qu'il n'est pas possible d'y intégrer le graphisme ni d'obtenir une certaine interaction entre les personnages, ce qui rend difficile de les doter d'une vie propre.

État d'âme

La famille des microprocesseurs 68000 de Motorola est apparue sur le marché en 1982. C'est le successeur de grande capacité du célèbre microprocesseur sur 8 bits, le 6809.

Le 68000 ressemble beaucoup au MC6800 qui est antérieur et toujours très répandu, notamment pour les périphériques tels que les contrôleurs évolués. Cela signifie que le plus puissant 68000 est simple à interfacer et dispose d'une compatibilité étendue avec de nombreux matériels. Parmi ceux-ci, des cartes entrée-sortie avec composants 6821 PIA (*Peripheral Interface Adaptator,* « adaptateur interface périphérique »), des unités de visualisation avec tubes 6845 CTRC (*Cathode Ray Tube Controllers,* « contrôleurs de tubes cathodiques »), des horloges avec rythmeurs programmables 6840 et des contrôleurs disque.

Le 68000 a également pour attrait d'offrir aux constructeurs une large gamme de bus de données et d'adresse. Ces derniers sont entièrement distincts les uns des autres, et chaque bit a sa propre broche — contrairement aux 8086, 8088 et Z8000, pour lesquels les broches sont multiplexées, les deux bus se partageant un jeu de broches. Les signaux sont séparés et décodés une fois arrivés à destination.

Aussi le processeur 68000 peut-il travailler aussi vite que le permet le système. Avec les nouveaux composants de RAM fonctionnant à 50 ou à 90 nanosecondes, cela peut aller jusqu'à supprimer les temps d'attente. Le microprocesseur le plus rapide de la famille des 68000 est le 68000L12 dont plusieurs opérations peuvent se faire à 14 MHz.

Sinclair a utilisé pour le QL le 68008, qui est le successeur du 68000. Il est, de manière interne, très semblable aux processeurs de la même famille, mais il est compatible avec les systèmes existants sur 8 bits, le bus de données étant sur 8 bits au lieu de 16. Du fait qu'il nécessite moins de broches, il n'en comporte que quarante.

Motorola doit prochainement produire un microprocesseur encore plus puissant, sur 32 bits, le 68020, qui sera sur quatre-vingt-seize broches. Le 68881 est un processeur de calcul mathématique de la virgule flottante, sur huit registres de 80 bits chacun. Il accroîtra considérablement la quantité de données effectivement traitées.

Beaucoup d'autres composants de la famille des 68000 comportent des fonctions d'entrée-sortie améliorant les performances de leurs prédécesseurs. Du point de vue du programmeur, le 68000 présente de nombreux avantages par rapport aux autres processeurs 16 bits. Ces avantages sont dus à la symétrie des registres de données et des registres d'adresse, et à la richesse du jeu d'instructions.

Cela dit, le 68000 n'est pas pour autant parfait. D'abord il y a cette distinction entre les deux types de registres, alors qu'ils sont de la même taille (32 bits) et sont gérés de manière similaire par les mêmes instructions. Le résultat est qu'il est souvent nécessaire de déplacer des données depuis un registre d'adresse sur un registre de données, les traiter, et les remettre sur le registre d'adresse. Cela aurait été plus facile si Motorola avait fait les registres aussi bien pour les données que pour les adresses.

Ensuite, certaines instructions se recoupent. Ce n'est pas grave puisque cela résulte des modes d'adressage se recoupant eux-mêmes. L'explication est la suivante : souvent très différents, ces modes d'adressage aboutissent au même résultat.

De manière générale, les microprocesseurs de la famille 68000 font des unités centrales de grande capacité, rapides et efficaces, et de plus en plus répandues. Le 68000 équipe le Lisa et le Mackintosh d'Apple, le QL de Sinclair et de nombreuses autres machines de gestion. Le 68000 offre, à un coût raisonnable, des ressources qui auraient coûté une fortune il y a seulement quelques années. La famille des 68000 rencontre autant de succès que le Z80 ou le 6502.

Processeurs vedettes

Le 6502
Développé par MOS Technology, il est avec le Z80 une des constantes du marché. Il utilise un bus d'adresse sur 16 bits, et un bus de données sur 8 bits. L'organisation de ses registres est un de ses points forts. Il existe un seul accumulateur, mais l'ensemble de la page mémoire 0 peut servir à des registres d'utilisation générale.

Le 68000
Motorola a revu le 6800 en 6809, mais trop tard pour s'assurer une part significative du marché des 8-bits. C'est pourquoi Motorola s'est tourné vers des processeurs plus puissants. Le 68000, 16-32 bits, utilise de nombreux composants de base de la famille 6502-6800, et est construit autour de huit registres de données à 32 bits, et de sept registres d'adresse également à 32 bits.

Le Z80
Théoriquement plus puissant que le 6502 de MOS, le Zilog 80 utilise des structures d'adresse et de données similaires, mais avec davantage de registres. Il comprend douze registres d'utilisation générale sur 8 bits, et deux registres d'index sur 16 bits. Son jeu d'instructions est également beaucoup plus étendu. Son atout principal par rapport au 6502 a sans doute été d'être compatible avec le système d'exploitation CP/M. (Cl. Kevin Jones.)

Premiers soins

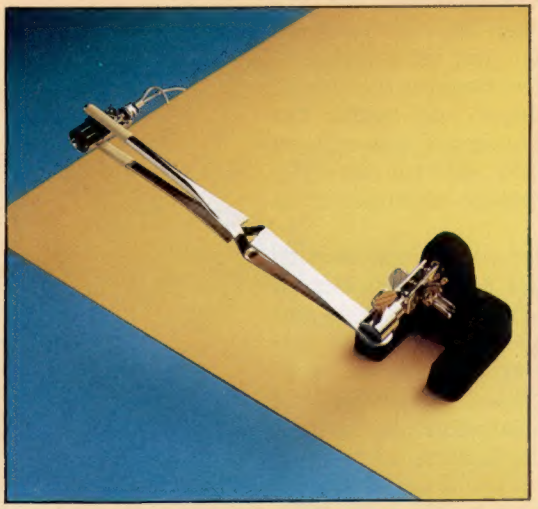
Les micro-ordinateurs et la haute-fidélité ont bien des choses en commun. Cette série d'articles permettra au débutant d'effectuer lui-même certains petits travaux, qui, autrement, se révéleraient très onéreux. Aujourd'hui : la soudure.

La soudure, comme le brasage, consiste à joindre, grâce à un alliage métallique aisément fusible (en électronique, on utilise le plomb et l'étain), deux objets de métal, que l'on chauffe d'abord à une température supérieure au point de fusion de cet alliage (280 °C), que l'on étale ensuite sur eux. Puis on les met en contact, et on interrompt le chauffage.

Un fer à souder mis au contact d'un fil d'alliage le fait fondre presque aussitôt. Toutefois, si le métal en fusion est déposé sur des pièces à joindre restées froides, il se solidifiera presque instantanément. C'est ce qu'on appelle une soudure « sèche ». Au mieux, elle ne tiendra pas. Au pire, elle ne donne qu'une connexion très médiocre, qui pourra même ne laisser passer le courant électrique que par à-coups. Un seul moyen d'empêcher cela : chauffer d'abord les deux pièces, jusqu'à ce que le fil à souder fonde par simple contact.

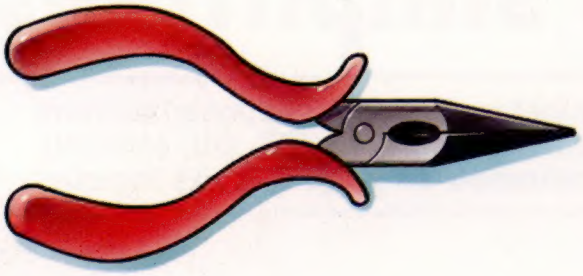
Une bonne prise

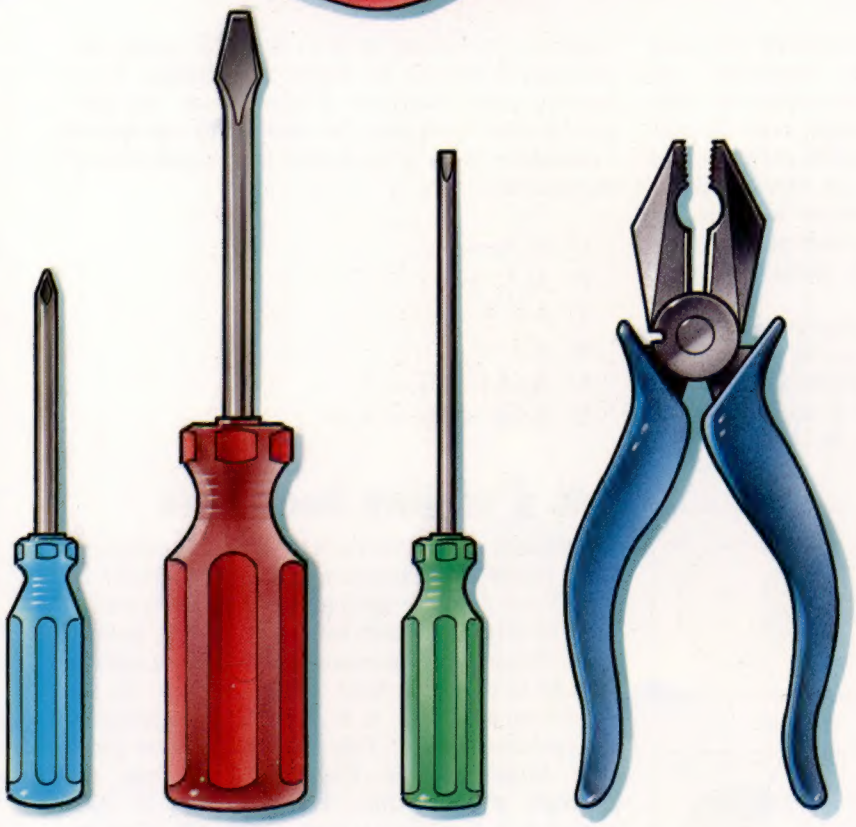
Si les micro-ordinateurs font désormais partie de notre vie quotidienne, c'est aussi à cause de leur taille réduite. Leurs composants sont minuscules et les manipuler est parfois difficile. Des pinces et des étaux légers, de prix raisonnables, vous y aideront. Si vous trouvez la prise insuffisante, servez-vous de chatterton. (Cl. Ian McKinnell.)



Plusieurs fers au feu
Il faut d'abord se souvenir que les composants électroniques sont très sensibles à la surchauffe. On en tiendra compte en choisissant un fer à souder (dimension de la pointe chauffante, diamètre du fil à souder). De façon générale — ainsi pour des raccords électriques ou de simples connexions — ce problème est cependant moins important. Un fer à souder de 15 ou 25 W et un fil d'alliage de 1,5 mm de diamètre se révéleront adaptés à la plupart des tâches.

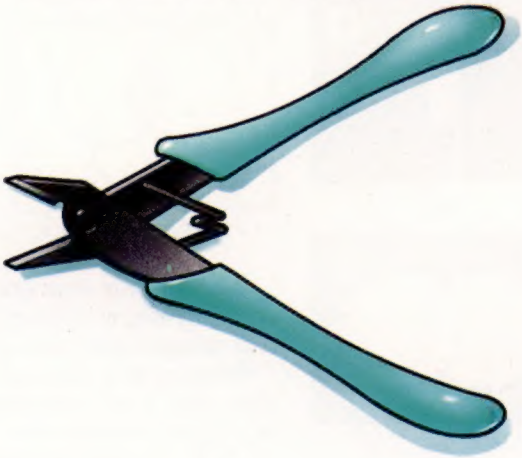
Ne pas perdre le fil
Chaque fois qu'on a affaire à des fils électriques, il faut d'abord les dénuder, puis, le travail accompli, les recouvrir de façon nette et propre. Une pince à dénuder comme celle-ci est très bon marché ; on peut la régler pour qu'elle ne morde que jusqu'à une certaine profondeur.





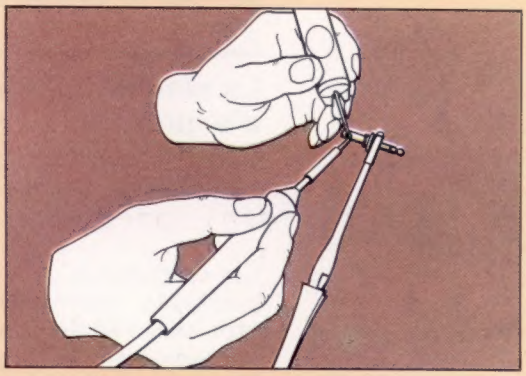
Un tournevis adapté
On distingue deux variétés de tournevis : plat et cruciforme. Ce dernier comprend d'ailleurs deux modèles, mais ils sont utilisés indifféremment sur la plupart des micro-ordinateurs. Sans doute aurez-vous besoin des uns et des autres, dans des tailles différentes.

Les pinces
Le débutant peut se satisfaire de deux modèles différents : la pince universelle (ci-dessus), indispensable pour tous les travaux en force et qui peut également servir à couper les fils électriques ; et la pince à long bec (en haut), irremplaçable quand on doit accomplir des tâches plus délicates. (Cl. Michael Brownlow.)

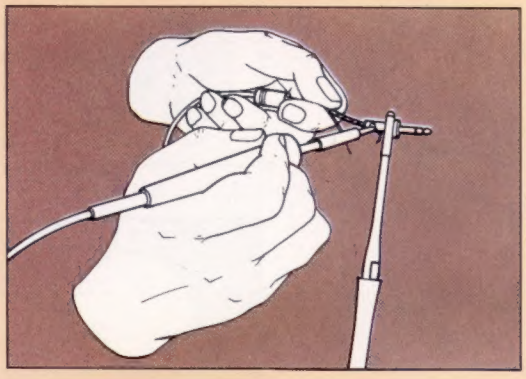


Petites coupures
Pour couper des fils électriques on emploie des pinces comme celle-ci, ou un modèle un peu différent, dont les mâchoires sont perpendiculaires au plan des poignées. En cas d'urgence, vous pourrez toujours utiliser un coupe-ongles, mais n'espérez pas pouvoir vous en servir encore par la suite...

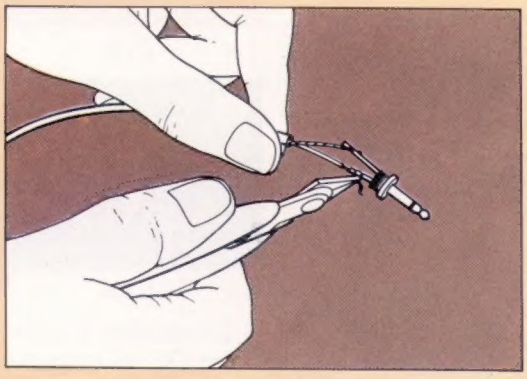
Une soudure simple



Dénudage et étamage
Lorsqu'on veut raccorder une fiche et un fil électrique, il faut d'abord dénuder celui-ci en enlevant la gaine isolante. Voyez large et sectionnez plus de gaine qu'il n'est nécessaire ; une fois l'opération terminée, vous supprimerez toute longueur de fil superflue. Maintenez le câble électrique dans l'étau, chauffez-le avec le fer, et appliquez-y la soudure. Quand elle commence à fondre, étendez-la proprement, à l'aide du fer, sur toute la surface dénudée. Ce procédé s'appelle l'étamage, et simplifie considérablement la soudure elle-même, puisque le composant fusible est désormais réparti sur les pièces à joindre. Répétez l'opération sur la fiche elle-même. Pendant qu'elle est encore assez chaude pour faire fondre le fil à souder, appliquez sur elle le câble étamé, retirez le fer dès que l'alliage commence à unir les deux éléments, soufflez pour refroidir et le tour est joué !



Ébarbage
Il faut maintenant ébarber tout élément superflu. Cela est valable aussi bien pour le fil que pour les broches de la fiche elle-même : mais il faut s'y prendre au tout dernier moment, c'est-à-dire lorsque le raccord a été testé. L'ébarbage est indispensable : fil ou broche peuvent en effet entrer en contact avec un autre composant, provoquant un court-circuit ou, pis encore, une de ces pannes intermittentes si exaspérantes. (Cl. Kevin Jones.)

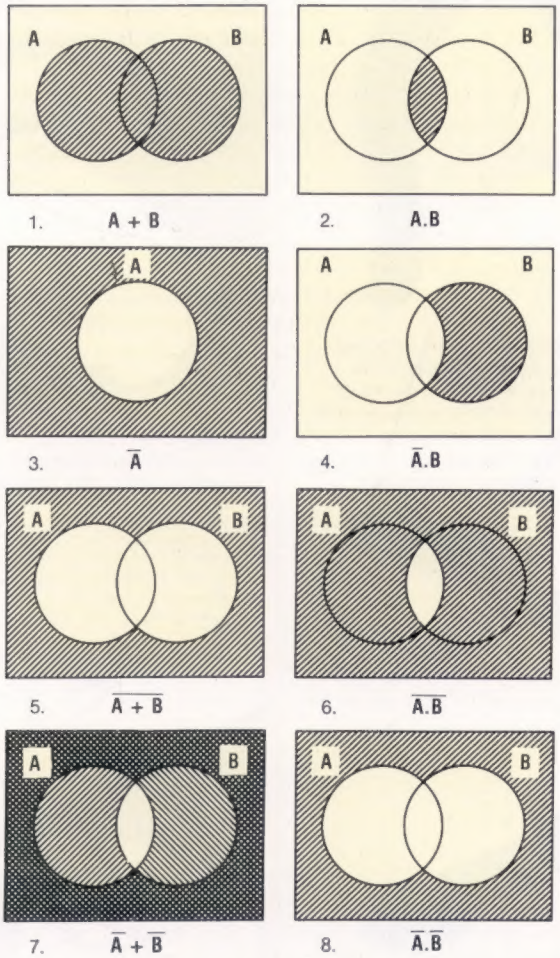


Formes simplifiées

La simplification de l'algèbre de Boole suppose l'écriture d'expressions avec moins d'opérateurs (ET, OU, et NON). C'est important pour concevoir de longs circuits logiques.

Les diagrammes de Venn fournissent une aide graphique appréciable pour simplifier les expressions booléennes, dans la mesure où elles les visualisent. La zone comprise dans un rectangle (symbolisée par 1 = identité ou ensemble universel) représente toutes les combinaisons possibles pour les valeurs vraies en données; les cercles représentent certaines combinaisons possibles. Voici des exemples de diagrammes de Venn ci-dessous.

La comparaison entre les diagrammes 5 et 7 montre d'un seul coup d'œil que NON(A OU B) est différent de NON(A) ou NON(B). Pareillement, les diagrammes 6 et 8 montrent que NON(A ET B) est différent de NON(A) ET NON(B).

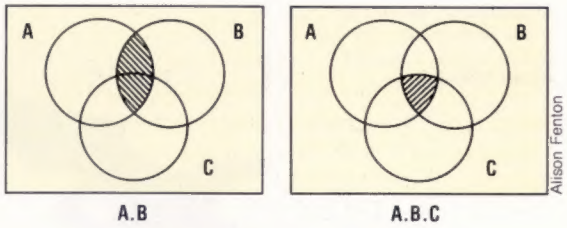


La manière la plus évidente de se représenter ET et OU est de penser à A.B comme à l'intersection de A et de B, et de penser à A + B comme à la fusion de A et de B. Il existe des relations évidentes en algèbre de Boole. Vous pouvez pour chacune d'elles tracer un diagramme de Venn pour les vérifier (0 représente l'ensemble vide, c'est-à-dire une combinaison impossible).

1° A.A. = A
2° A.$\overline{A}$. = 0
3° A.0 = 0
4° A.1 = A
5° A.(A + B) = A
6° A.($\overline{A}$ + B) = A.B

Lois d'algèbre booléenne

Le concept de *dualité* est très utile et... curieux! Il est fondé sur la symétrie des opérateurs ET et OU. Pour écrire le double d'une relation vraie, il suffit d'inverser tous les OU et les ET, les 0 et les 1. Prenons par exemple la cinquième opération de la liste plus haut : A.(A + B) = A. Le double en est : A + A.B = A. Cette expression est également vraie. Elle illustre un autre principe important de l'algèbre booléenne, le *principe d'absorption*. En regardant le diagramme de Venn pour cette expression, il est en effet évident que le terme A.B est entièrement « absorbé » par A. Ce principe peut être étendu à un exemple sur trois variables tel que : A.B + A.B.C. = A.B. Les deux diagrammes de Venn suivants confirment cette règle.



Alison Fenton

Vous pouvez vous exercer à écrire les doubles des cinq autres relations spéciales, et à tracer leur diagramme pour en confirmer la validité.

Examinons à nouveau les diagrammes donnés plus haut. Les diagrammes 5 et 8 montrent que la relation suivante est toujours vraie : $\overline{A + B} = \overline{A}.\overline{B}$. La comparaison des diagrammes 6 et 7 met en évidence la relation suivante : $\overline{A.B} = \overline{A} + \overline{B}$. Ces deux relations sont appelées *lois de Morgan*. Elles sont applicables à des expres-

sions plus complexes, à trois variables, par exemple : $\overline{A + B + C} = \overline{A}.\overline{B}.\overline{C}$ et $\overline{A.B.C} = \overline{A} + \overline{B} + \overline{C}$.

Les lois de Morgan peuvent également s'appliquer par étapes :

$(\overline{A + B}).\overline{C}$

$= \overline{A}.\overline{B}.\overline{C}$ (selon la loi de Morgan sur les parenthèses)

$= \overline{A + B + C}$ (selon l'utilisation recombinée de la loi de Morgan).

Il existe trois règles communes à l'algèbre normale et à l'algèbre de Boole. La loi associative permet de supprimer des parenthèses :

$(A.B).C = A.(B.C) = A.B.C$

$(A + B) + C = A + (B + C) = A + B + C$

L'ordre d'écriture peut être changé en vertu de la loi commutative :

$A.B = B.A$

$A + B = B + A$

La loi distributive permet de « distribuer » une opération :

$A.(B + C) = A.B + A.C$

Exemples de simplification

1° Simplifier $(\overline{A + B} + \overline{A}.B).B$

$= (\overline{A}.\overline{B} + \overline{A}.B).B$ (loi de Morgan)

$= \overline{A}.\overline{B}.B. + \overline{A}.B.B$ (loi distributive)

$= 0 + \overline{A}.B$ ($\overline{B}.B. = 0$ et $B.B = B$)

$= \overline{A}.B$

2° Simplifier $\overline{A}.\overline{B}. + \overline{A}.B + A.B$

$= \overline{A}.(\overline{B} + B) + A.B$ (loi distributive)

$= \overline{A} + A.B$ ($\overline{B} + B = 1$)

$= \overline{A} + B$ (double de la relation 6).

3° Simplifier $\overline{\overline{A} + B} + \overline{\overline{A} + \overline{B}} + \overline{A} + B$

$= \overline{\overline{A}}.\overline{B} + \overline{\overline{A}}.\overline{\overline{B}} + \overline{A}.B$ (loi de Morgan)

$= A.\overline{B} + A.B + \overline{A}.B$ ($\overline{\overline{A}} = A$)

$= A.(\overline{B} + B) + \overline{A}.B$ (loi distributive)

$= A + \overline{A}.B$ ($\overline{B} + B = 1$)

$= A + B$ (double de la relation 6).

Porte logique XOU simplifiée

Nous avons vu dans le dernier numéro le circuit complet pour une porte logique XOU (OU exclusif). Reprenons-le :

ENTRÉE		SORTIE
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Nous avons déduit de la table que $C = \overline{A}.B + A.\overline{B}$. Il n'est pas possible de simplifier de manière significative cette expression, et un circuit à cinq portes logiques serait nécessaire pour l'implémenter. Mais il existe un autre moyen : C peut être considéré comme égal à 1 si A et B ne sont pas ensemble respectivement 1 ou 0.

Nous pouvons écrire C en algèbre booléenne :

$C = \overline{A.B + A.B}$

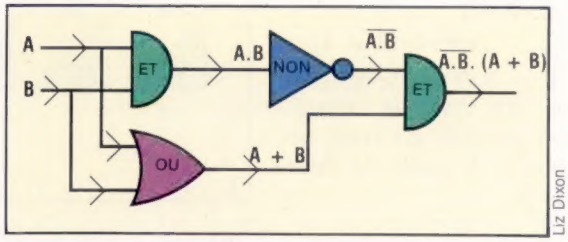
En utilisant plusieurs fois les lois de Morgan, nous pouvons simplifier le circuit :

$C = (\overline{A.B}).(\overline{A.B})$

et finalement :

$C = \overline{A.B}.(A + B)$

Cette expression ne prend que quatre portes logiques :



Un circuit additionneur complet

Nous avons déjà abordé l'addition binaire et élaboré un circuit simple destiné à additionner deux bits et à produire deux résultats, pour la somme et pour la retenue. Il s'agissait d'un circuit « demi-additionneur ». Si nous appelons X et Y respectivement la première et la deuxième entrée, nous pouvons vérifier d'après la table de vérité pour ce type de circuit que le résultat (la somme) peut s'écrire sous la forme de l'expression suivante : $S = \overline{X}.Y + X.\overline{Y}$. Après simplification par les lois de Morgan : $S = \overline{X.Y}(X + Y)$. La retenue est simplement : $C = X.Y$.

En arithmétique binaire, il y a trois chiffres à additionner, et donc trois colonnes pour la somme. Outre les deux chiffres à additionner, il y a la retenue de la colonne précédente. Pour reproduire le processus d'addition binaire, il faut concevoir un circuit à trois entrées et deux sorties. Si nous appelons la retenue de la colonne précédente P, la table de vérité pour un circuit additionneur complet sera :

ENTRÉES			SORTIES	
P	X	Y	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

En prenant le cas où S = 1, il est possible d'établir une expression pour S à partir de la table de vérité :

$S = \overline{P}.\overline{X}.Y + \overline{P}.X.\overline{Y} + P.\overline{X}.\overline{Y} + P.X.Y$

Expression simplifiable :

$S = \overline{R}.(\overline{X}.Y + X.\overline{Y}) + P.(\overline{X}.\overline{Y} + X.Y)$

[loi distributive]

$S = \overline{P}.(\overline{X}.Y + X.\overline{Y}) + P.\overline{(\overline{X}.Y + X.\overline{Y})}$

[loi de Morgan]

De manière similaire, nous pouvons écrire une expression pour C à partir de la table de vérité :

$C = \overline{P}.X.Y + P.\overline{X}.Y + P.X.\overline{Y} + P.X.Y$

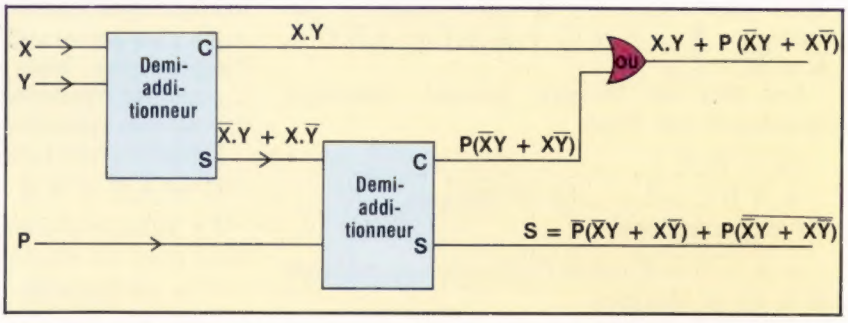
$C = X.Y.(\overline{P} + P) + P.(\overline{X}.Y + X.\overline{Y})$

[loi distributive]

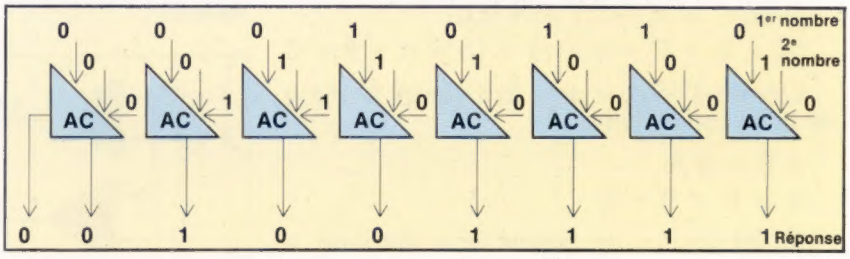
$C = X.Y + P.(\overline{X}.Y + X.\overline{Y})$

$(\overline{P} + P = 1)$

On remarque que $\overline{X}.Y + X.\overline{Y}$ est le résultat de la somme selon un circuit demi-additionneur. Il est ainsi possible de créer un circuit additionneur complet à partir de deux circuits demi-additionneurs.



L'exemple suivant indique comment huit circuits additionneurs complets se combinent dans le cadre de l'unité arithmétique et logique, pour effectuer l'addition binaire de deux nombres sur 8 bits :



EXERCICE 3

1° Simplifier les expressions suivantes :

a) $A.(\overline{A} + \overline{B})$

b) $X + Y.(X + Y) + X.(\overline{X} + Y)$

c) $\overline{P.Q + \overline{P}.Q + \overline{P}.\overline{Q}}$

d) $\overline{\overline{X} + Y.\overline{Z} + \overline{Z}.Y}$

2° Un système d'alarme de voiture comporte un commutateur et un détecteur sur les deux portes avant. La sirène d'alarme se déclenchera si l'on cherche à ouvrir l'une des deux portes (ou les deux) lorsque le système est commuté (branché). Tracez une table de vérité qui indique les trois entrées de données (la porte A, la porte B, le commutateur du système), et la sortie logique du circuit, le déclenchement ou le non-déclenchement de l'alarme. Utilisez cette table pour écrire une expression booléenne pour le déclenchement de l'alarme, et tracez le circuit logique du système d'alarme.

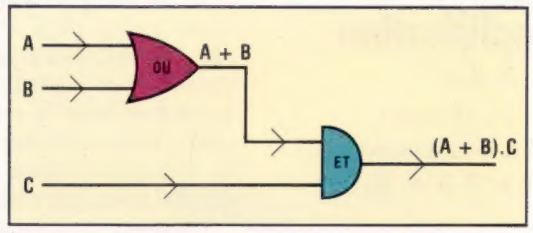
3° La lumière d'un escalier se commande depuis un commutateur à l'entrée, un autre au bas de l'escalier, et un troisième en haut. Réalisez un circuit logique approprié.

4° Vous êtes abandonné sur une île déserte avec deux autres personnes. L'une d'elles dit toujours la vérité, l'autre ment toujours. Pour votre survie, il faut absolument que vous sachiez laquelle dit la vérité. Pour y parvenir, vous devez imaginer des questions qui démasqueront le menteur. Tracez des tables de vérité pour envisager les réponses possibles selon la personne à laquelle les questions s'adressent. Voici un exemple :
« Dites-vous toujours la vérité ? »

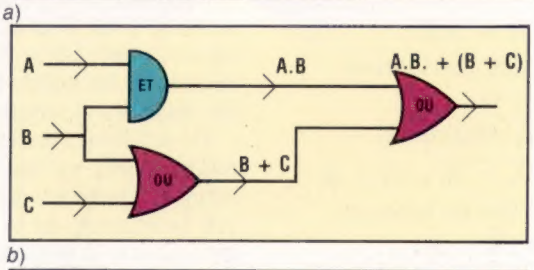
		RÉPONSE POSSIBLE	
		OUI	NON
IDENTITÉ POSSIBLE	menteur	1	0
	vrai	1	0

Réponses à l'exercice 2

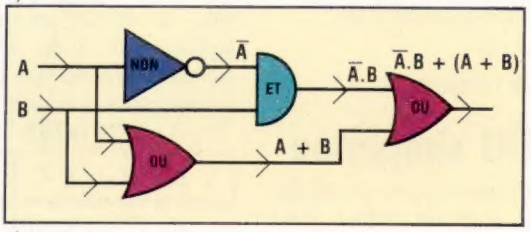
1°



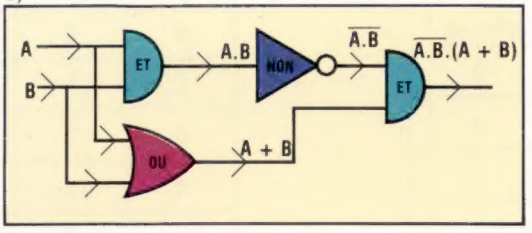
a)



b)



c)



d)

2° *a*) $C = \overline{A.B}$

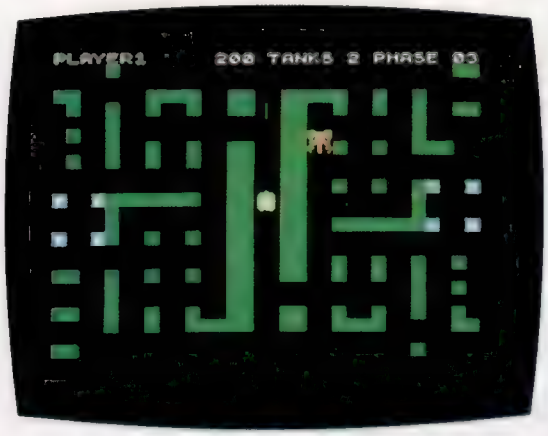
b) $S = \overline{A.B}\ (A + B)$

3° *a*) $X = (A + B).(B.C + C)$

b) $X = \overline{A.B} + \overline{B}$

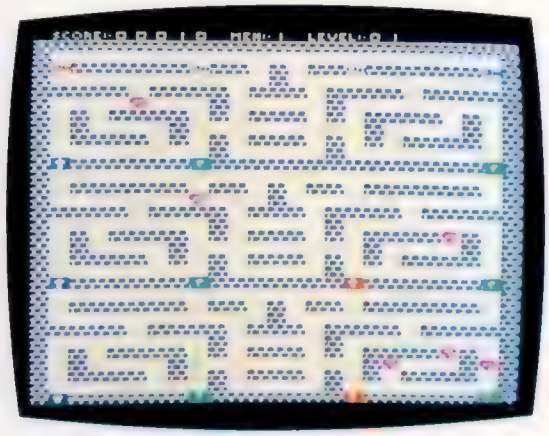
Labyrinthes

Avec les batailles spatiales, les jeux de labyrinthes sont également très prisés par le grand public. Rappelez-vous du succès de PacMan! En voici quatre autres.



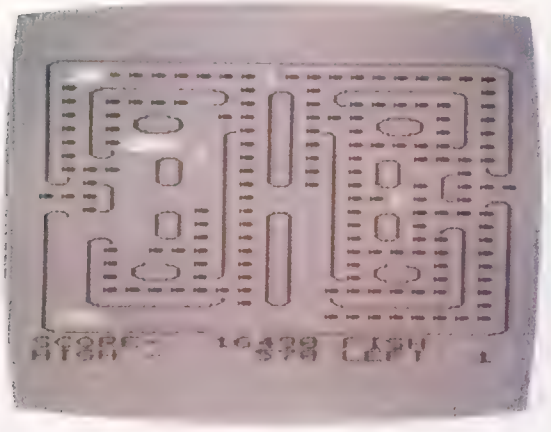
SPECTRUM BRAIN DAMAGE

C'est un jeu d'oracle qui se passe dans un labyrinthe et dont le principe de base est de tuer vos ennemis avant qu'ils ne vous tuent eux-mêmes. Vos adversaires sont le Panzer électronique, le Centurion et le Maraudeur, qui peuvent tirer dans toutes les directions et souvent avec des effets très dévastateurs pour votre propre tank.
Les graphiques ne sont pas de très bonne qualité, mais les effets sonores sont intéressants.



VIC-20 BEWITCHED

C'est un produit de la Société Imagine qui est très attractif. Le jeu consiste en un voyage dans un labyrinthe qui comporte des portes à serrures.
L'écran représente le labyrinthe divisé en trois sections. Les sections inférieures sont bloquées par quatre portes de couleurs différentes, fermées à clef et qui ne peuvent être ouvertes que par des clefs de même couleur qui se trouvent dans la première partie du labyrinthe, en haut de l'écran.



BBC FELIX AND THE FRUIT MONSTERS

Un labyrinthe relativement simple, dans le genre classique de PacMan. Il y a des pilules énergétiques dans des coins du labyrinthe qui vous permettent d'esquiver les monstres qui essaient de vous attraper, mais au lieu de les manger comme dans un PacMan classique, il faut déplacer les fruits autour du labyrinthe en n'en transportant qu'un seul à la fois. Un jeu moyen, dans l'ensemble, qui présente un défaut majeur à signaler : le concepteur n'a pas prévu d'option de changement des touches de contrôle, ce qui peut être très gênant!

ATARI WAY OUT

Un jeu de labyrinthe extrêmement attractif par ses effets visuels et sonores et par les simulations en trois dimensions très bien réussies. On peut se déplacer dans 26 labyrinthes différents, dont la liste est affichée au début du jeu et que l'on sélectionne par un code. Le joueur dispose d'un compas pour se diriger et d'un plan général du labyrinthe pour le choix d'une direction. Mais chacun des labyrinthes est habité par le « cleptangle », entité mystérieuse qui ressemble étrangement à la fameuse « dalle noire » de *2001, l'Odyssée de l'espace*.

Jusqu'au bout

Peu de constructeurs d'ordinateurs espèrent vendre un million d'exemplaires d'un modèle donné; c'est ce que Sinclair a fait avec le Spectrum. Celui-ci en est actuellement à sa troisième version.

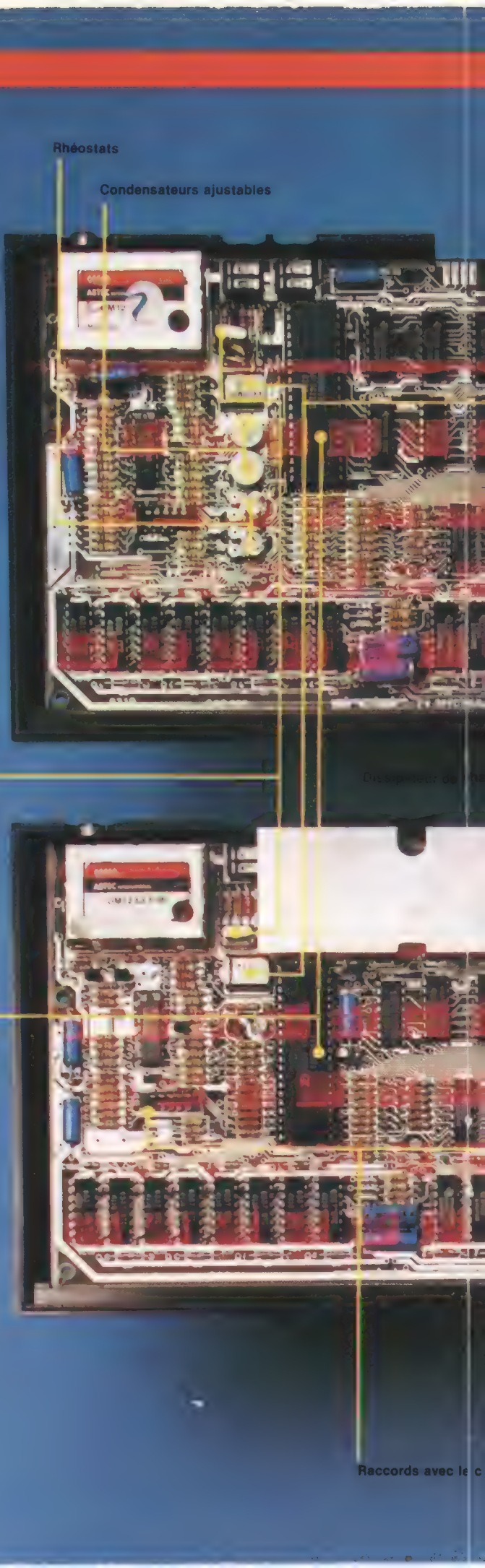
S'agissant de la construction elle-même, c'est avant tout le clavier que l'on remarque. Il est bel et bien de type QWERTY, mais les ressemblances avec une machine à écrire s'arrêtent là. Chacune des quarante touches repose sur une membrane, qui donne à la frappe un certain jeu; l'ensemble paraît mou et même « spongieux ».

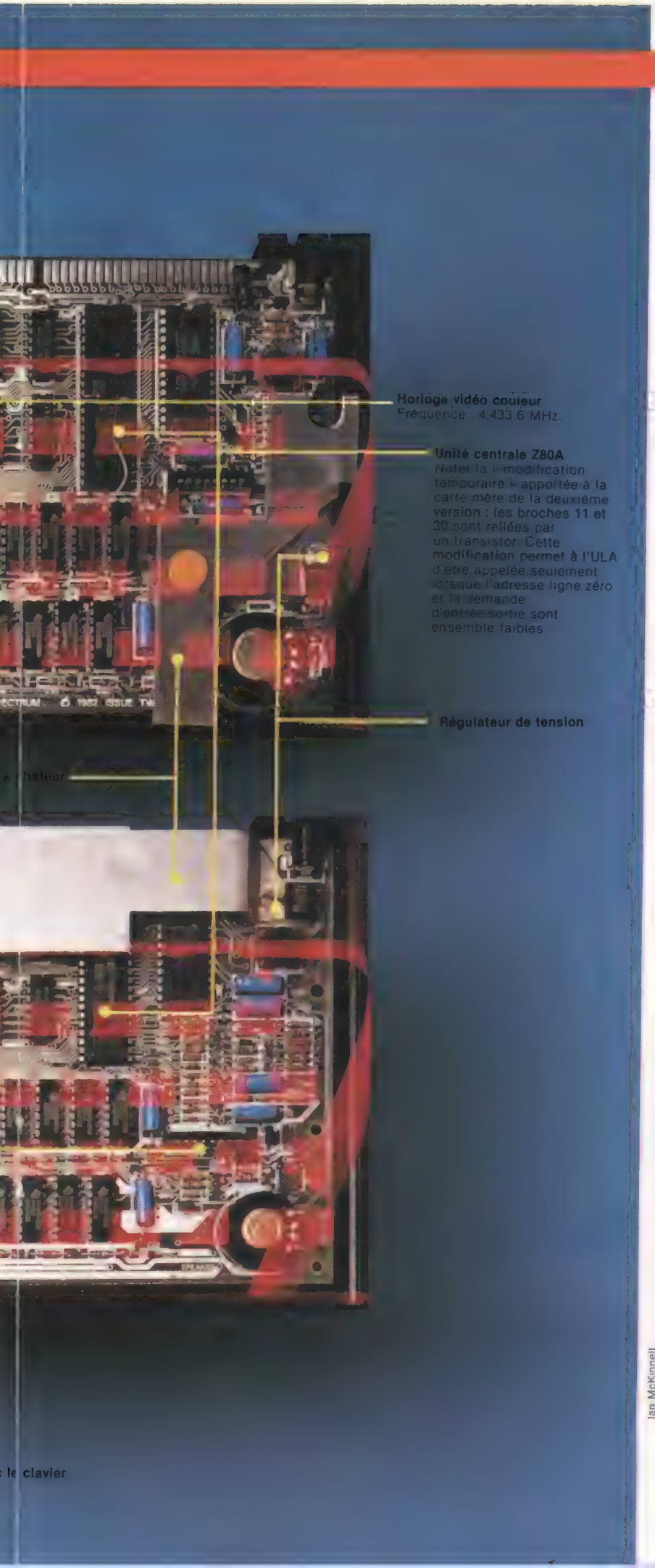
Un bus d'extension permet au Spectrum de se raccorder à l'imprimante ZX (prévue à l'origine pour le ZX-81), aux interfaces ZX 1 ou 2, ou aux trois en même temps. Il existe aussi des prises EAR et MIC qui permettent l'enregistrement des programmes sur cassette. Le chargement des logiciels n'est guère satisfaisant. Ces deux opérations, une fois achevées sans encombre, entraînent l'apparition, en bordure d'écran, de lignes bleues et jaunes peu rassurantes; mais l'enregistrement, par exemple, nécessite d'abord la mise hors service de la prise EAR. Il n'existe pas non plus de bouton de remise à zéro, ce qui implique qu'en cas de blocage ou de « plantage » du système il faut couper le courant, ce qui peut à la longue se révéler mauvais pour les connexions. Heureusement, un certain nombre de firmes indépendantes fournissent des dispositifs (interrupteurs Save or Load et Reset) qui résolvent ces problèmes.

Sinclair Research paraît, là comme ailleurs, avoir été largement distancé par d'autres; mais c'est une politique qui semble délibérée : négligeant les petits problèmes de maintenance, la firme, grâce aux profits que lui ont valu les ventes, peut se consacrer à d'autres produits, comme le QL. Elle a d'ailleurs, malgré tout, mis au point les Microdrive et l'interface ZX 1, qui permettaient d'obtenir une capacité de mémoire théorique de 680 K, de relier en réseau soixante-quatre Spectrum et d'employer un port RS232. Il y a eu depuis l'interface ZX 2, qui autorise le raccord de deux manches à balai, et l'accès aux logiciels inscrits en ROM.

Sinclair Research semble également avoir abandonné à autrui la production de logiciels, et se borne à donner sa garantie, sous forme d'approbation officielle.

L'évolution du Spectrum
De 1982 (date de sa sortie) au début de 1984 (lancement du QL), le ZX Spectrum de Sinclair s'est vendu à plus d'un million d'exemplaires et a connu trois versions différentes de la carte mère (le principal circuit imprimé, qui accueille les composants essentiels). La première se limite aux soixante mille premiers exemplaires et n'est donc pas très répandue. Les deux suivantes présentent deux grandes différences : tout d'abord, on pouvait procéder sur la deuxième version à un réglage fin du signal vidéo en manipulant les deux condensateurs ajustables et les deux rhéostats signalés sur le schéma. Ensuite, la « modification temporaire » apportée au microprocesseur de la deuxième version avait été menée à bien lorsque la troisième version fut mise en vente. Le dissipateur de chaleur a été déplacé, le circuit imprimé chargé de réguler la tension ayant été installé plus près de la prise d'alimentation.





Ian McKinnell

SINCLAIR SPECTRUM

PRIX

1 900 F (16 K).
3 000 F (48 K).

DIMENSIONS

233 × 144 × 30 mm.

UNITÉ CENTRALE

Z80A.

CAPACITÉ MÉMOIRE ET VITESSE

16 et 48 K.
3.5 MHz.

CARACTÉRISTIQUES ÉCRAN

24 lignes de 32 caractères. Graphisme haute résolution (256 × 192). 16 caractères graphiques prédéfinis, plus 21 caractères programmables. Huit couleurs, clignotement, deux niveaux de luminosité. Couleur de la bordure d'écran indépendante.

INTERFACES ET PORTS

Bus d'extension. Prises télévision et cassette pour stockage des programmes.

LANGAGES DISPONIBLES

BASIC et assembleur du Z80. Il est aussi possible d'installer FORTH, LOGO, Micro-PROLOG et bien d'autres.

CLAVIER

40 touches mobiles à membrane, aux standards ASCII. Une touche peut avoir jusqu'à huit fonctions différentes, en conjonction avec les touches SHIFT.

DOCUMENTATION

L'appareil est accompagné d'une brochure d'introduction et du manuel proprement dit, qui comporte une étude du fonctionnement du Spectrum, ainsi que des précisions sur le BASIC Sinclair.

POINTS FORTS

D'un prix raisonnable pour ses 48 K, le Spectrum est un outil idéal pour un débutant. De très nombreuses firmes ont produit pour lui énormément de matériels et de logiciels.

POINTS FAIBLES

Le clavier est astucieusement conçu, mais ne peut se comparer à celui d'une machine à écrire. L'affichage d'écran est de type très particulier, et peut poser des problèmes aux débutants.

Demandez les programmes

Il existe de très nombreux jeux destinés au Spectrum, mais aussi un assez grand nombre de programmes de gestion (financière ou familiale). Ils sont vendus à des prix généralement très raisonnables. Avec l'apparition des Microdrive, il est désormais tout à fait possible de faire tourner sur Spectrum d'importantes bases de données, des tableurs ou des logiciels de traitement de texte.
Psion, un des plus proches collaborateurs de Sinclair Research pour ce qui est de l'élaboration des programmes, est la firme à qui on doit nombre de ces logiciels. Elle a ainsi lancé une série qui a eu beaucoup de succès : VU-CALC, VU-File, VU-3D.
La compagnie Microl a, en dépit de la médiocrité du clavier du Spectrum, mis sur le marché The Word Processor, qui permet de stocker jusqu'à dix pages de texte, et qui offre bien des particularités d'habitude réservées aux traitements de texte plus élaborés (ainsi la possibilité de fusionner des fichiers).

Commandes de disque

Commodore fabrique depuis plusieurs années toute une gamme de lecteurs 5 pouces. Ils sont tous « intelligents », et comportent donc leur propre microprocesseur et leur mémoire vive (RAM) associée.

Le problème essentiel des lecteurs dits « intelligents » est leur coût. Après le lancement du micro-ordinateur Vic-20, Commodore proposa une version bon marché de ses lecteurs célèbres, PET, sous la forme d'un lecteur à une disquette, le Vic-1540. Le Commodore 64 a les mêmes caractéristiques, quant à l'accès, que le lecteur 1540. Cependant, certaines différences ont rendu nécessaire l'emploi d'une commande POKE avant et après l'utilisation. Le remplacement du lecteur 1540 par le lecteur 1541 a permis d'éliminer cet inconvénient par la modification du système d'exploitation. Cette nouvelle version est compatible avec le Vic-20 et le Commodore 64. Pour des raisons de commodité, nous nous référerons aux deux lecteurs en partant du même nom, 1541.

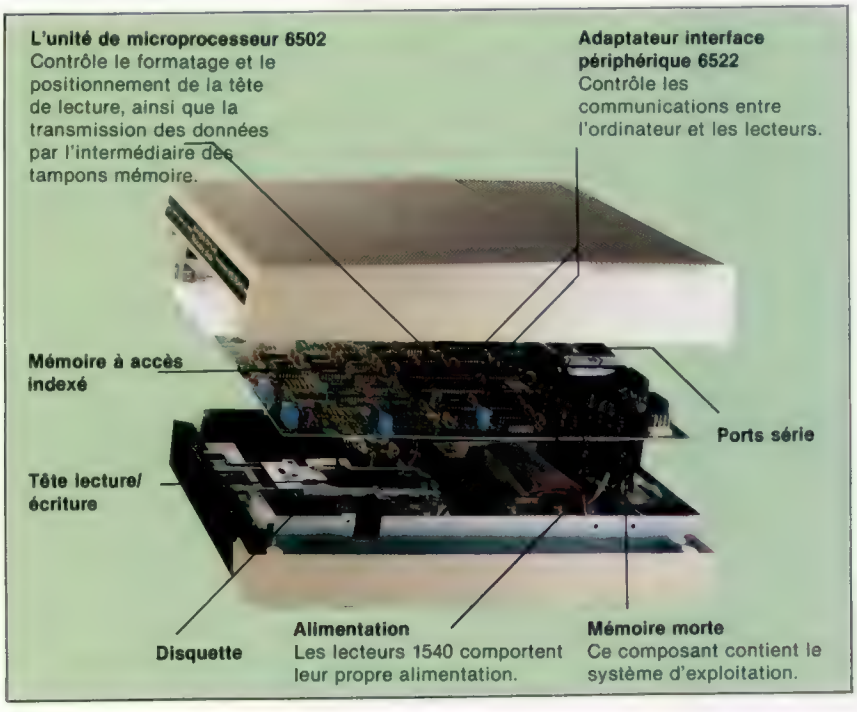
Le contrôle du 1541 passe par un microprocesseur 6502, deux adaptateurs interface polyvalents 6522, une mémoire vive de 2 K et une mémoire morte de 8 K qui contient le microprocesseur. Ce dernier est très puissant et permet de programmer des routines complexes de gestion de fichiers programmes, de fichiers séquentiels et indexés, avec des procédures de gestion d'erreurs sophistiquées. Le contrôle de l'ordinateur est assuré par une version série de l'interface IEEE488. Celle-ci comporte les mêmes commandes que son puissant équivalent parallèle qui contrôle les autres périphériques Commodore. En outre, l'interface série IEEE488 permet le chaînage en marguerite des interfaces. Ainsi, un lecteur de disque peut transmettre des fichiers en impression pendant que l'ordinateur effectue d'autres tâches. Cela s'obtient par des commandes associées avec des numéros appropriés de fichiers logiques et d'unités de disque.

Les disquettes sont formatées sur 35 pistes sur une seule face et chaque face comporte des secteurs (21 secteurs pour la piste la plus externe et 17 pour la piste la plus interne). Chaque secteur comporte un bloc de données de 256 octets avec en plus des données portant sur la synchronisation, le nom et le bit de parité (vérification).

Une disquette permet de stocker 683 blocs, dont 664 sont disponibles pour l'utilisateur. Ce qui donne une capacité maximale de 170 K approximativement, selon le type de fichier. Le système d'exploitation gère la répartition des données par une carte de disponibilité des blocs et un répertoire. La carte est sur le secteur 0 de la piste 18 et comporte 144 octets. Elle indique les blocs utilisés et les blocs disponibles. Le répertoire commence sur le secteur 1 de la piste 18 et comprend une liste maximale de 144 fichiers répartis par leur nom. Ce dernier indique le type du fichier et son nombre de blocs. A chaque mouvement de données, la carte de disponibilité des blocs et le répertoire sont mis à jour.

Malgré son prix d'achat élevé, la souplesse d'utilisation du système de lecteur de disque Commodore en fait un investissement justifié. Le stockage en mémoire de masse qu'il permet peut être incorporé à un système de gestion des périphériques et ne pèse pas sur la mémoire de l'ordinateur ou sur le minutage du processeur. Cependant, du fait de l'interface série, le lecteur 1541 est relativement lent. Commodore n'indique même pas une vitesse de transmission des données ou un temps d'accès moyen. Bien que la vitesse de stockage soit cinquante fois supérieure à celle d'une unité de cassette, le lecteur 1541 est vraisemblablement le plus lent des lecteurs pour ce qui est de l'accès. Si une grande capacité de stockage est nécessaire, une interface connectable sur le port série peut être achetée, qui imitera le port parallèle des micros de type PET. Cela n'accélère pas particulièrement les opérations, mais vous permet de connecter tous les périphériques PET sur un Vic-20 ou un Commodore 64.

Lecteur de disque Commodore
Disponible sur le VIC-20 et sur le Commodore 64, et prévu en série pour le futur ordinateur portable SX 64, le lecteur 1540 est du type « intelligent », qui ne dépend pas de l'unité centrale ou de sa mémoire. Il comporte son propre microprocesseur : le système d'exploitation mémoire 6502 qui équipe les ordinateurs eux-mêmes. (Cl. Ian McKinnell.)



Les commandes disque du Commodore

Le système d'exploitation du Commodore comporte une large gamme de commandes d'élaboration de programmes de gestion de fichiers indexés, en dehors des procédures habituelles de gestion de programme et de données. Ces commandes s'utilisent de la manière indiquée ci-dessous. Pour chaque cas, 8 est le numéro de lecteur.

SAVE
Crée des fichiers programme dont le nom porte sur seize caractères, et qui peuvent comporter soit des programmes soit des données.
La syntaxe est la suivante :

SAVE « NOM DU FICHIER », 8

LOAD
De syntaxe :

LOAD « NOM DU FICHIER », 8

Copie le fichier programme spécifié, en mémoire vive, depuis les positions de poids faible. La commande.

LOAD « NOM DU FICHIER » 8, 1

recopie le fichier indiqué sur les positions mémoire dont il avait été sorti par SAVE.

LOAD « S », 8

Copie le répertoire disque en mémoire utilisateur. Il peut ensuite être listé (LIST) comme un programme BASIC, et il contient :
— nom du disque;
— identificateur disque sur deux caractères;
— maximum de 144 noms de fichiers;
— fichiers de type programme ou séquentiel;
— longueur en blocs pour chaque fichier;
— nombre de blocs disponibles.

VERIFY
De syntaxe :

VERIFY « NOM DU FICHIER », 8

Compare le contenu du fichier avec celui du fichier couramment contenu en mémoire utilisateur. Génère un message d'erreur s'ils sont différents. Est utilisé pour vérifier si les fichiers ont été sauvegardés (SAVE) convenablement.

OPEN
Établit un canal dit « numéro de fichier logique », compris entre 1 et 255. Il est possible d'ouvrir jusqu'à dix numéros de fichiers logiques en même temps. OPEN crée également une adresse secondaire qui détermine le mode d'action du périphérique. La seule adresse secondaire pour un lecteur est 15. Elle donne accès au canal prioritaire "command". La syntaxe d'OPEN est la suivante :

OPEN LFN, 8, SA

CLOSE
De syntaxe :

CLOSE LFN

Ferme le fichier logique spécifié. Les fichiers logiques doivent toujours être fermés quand on n'a plus besoin d'eux.

PRINT #, INPUT # AND GET #
PRINT # fonctionne de la même manière que PRINT, à la différence près que les données sont sorties sous la forme d'un fichier séquentiel, vers le fichier logique OPEN, plutôt que vers l'écran. De syntaxe :

PRINT # LFN, « DATA » ou
PRINT # LFN, A$, B$,...

De la même manière, INPUT # et GET # acceptent en lecture des fichiers séquentiels. INPUT # restitue des chaînes de données, mais ne sera efficace que si les chaînes sont séparées par des points-virgules ou par des virgules, faute de quoi les chaînes seraient considérées comme une seule chaîne. GET # restitue les données octet par octet, y compris les points-virgules et les virgules. Ce qui est très utile lorsque le contenu d'un fichier est inconnu ou ne comporte pas de séparateurs. Voici des exemples de syntaxes :

INPUT # LFN, A$, B$.
GET # LFN, A$, B$...

Lorsque PRINT # est utilisé en conjonction avec un fichier logique ouvert (OPEN) vers un canal de commande (c'est-à-dire OPENLFN, 8, 15), sous la forme suivante :

PRINT # LFN, 8, 15, « chaine de commande »

PRINT # devient alors la commande gestion de disque la plus puissante possible. Les chaînes de commande servent à implémenter les commandes de maintenance de disque et les commandes de gestion de fichiers à accès direct (relatif, REL).

Commandes de maintenance de disque

Utilisées sur le canal de commande, avec PRINT # ou OPEN #, et selon la syntaxe appropriée, elles ont les fonctions suivantes :

NEW
Formate et donne un nom à la disquette; établit la table de disponibilité de blocs, et le répertoire.
Donne une valeur à l'identificateur de disque sur deux caractères; commande : « N : DISKNAME, DI »

INITIALISÉ
Ajuste la table de disponibilité des blocs présente en RAM disque avec celle qui se trouve sur le disque.
Commande : « I »

VALIDATÉ
Supprime les blocs alloués par les commandes REL qui ne figurent pas au répertoire, ainsi que les fichiers non fermés (CLOSE) après avoir été écrits.
Établit une nouvelle table de disponibilité des blocs.
Commande : « V »

RENAME
Change le nom du répertoire d'un fichier.
Commande : « R : NOUVEAU NOM = ANCIEN NOM »

SCRATCH
Supprime le fichier demandé du disque et du répertoire.
« S : NOM DU FICHIER 1, NOM DU FICHIER 2, ... »

COPY
Fait une copie du fichier demandé, sur le disque même.
Commande : « C : NOM COPIE = NOM ORIGINE »
Réunit les fichiers séquentiels en un seul et même fichier séquentiel sur le même disque, « concatène » les fichiers.
Commande : « C : NOM CONCA = NOM 1, NOM 2, ... »

Vérification des erreurs
Le panneau frontal du lecteur de disque 1541 comporte une diode verte « lecteur allumé », et une diode rouge indiquant :
• *On.* lecture ou écriture en cours.
• *Off.* dans l'attente d'instructions.
• *Clignotement.* le système d'exploitation a détecté une erreur.
Pour découvrir la nature de l'erreur, il faut lire le canal d'erreur du système d'exploitation. Le programme suivant imprime les codes d'erreur générés par le système d'exploitation. On les trouvera dans le manuel utilisateur.

L'ABC du BBC

Nous poursuivons notre exposé du BASIC résident BBC. Ce langage est aussi riche que la machine elle-même, et son apparition sur le marché a été aussi décisive que celle du BBC.

L'utilisation de ELSE dans l'instruction IF...THEN, sous la forme IF...THEN...ELSE, permet de supprimer la plupart des instructions GOTO. En effet, avec cette forme on peut traiter dans une même instruction l'état *vrai* ainsi que l'état *faux* de l'expression conditionnelle.

Par exemple :

```
1500 IF TEST>0 THEN GOTO 1800
1600 PRINT « VALEUR HORS ECHELLE
1700 GOTO 1900
1800 PRINT « SANS PROBLEME »
1900 NEXT L
```

pourra être avantageusement remplacé par :

```
1500 IF TEST>0 THEN PRINT « SANS PROBLEME »
     ELSE PRINT « VALEUR HORS ECHELLE »
1900 NEXT L
```

GOSUB prend généralement un numéro de ligne pour argument. Cela présente deux inconvénients : d'abord (et prenons un exemple) GOSUB 1000 ne donne aucune indication sur le sous-programme à la ligne 1000; ensuite, les renvois à des numéros de ligne rendent le programme difficile à lire et éventuellement à fusionner.

GOSUB, comme GOTO, est relativement lent à exécuter, du fait que la ligne spécifiée doit être cherchée à chaque occurrence de l'instruction.

Le BASIC du micro-ordinateur BBC apporte des réponses à ces problèmes. Ses fonctions et procédures sont semblables à des blocs de code du genre sous-programme. Elles sont appelées par l'intermédiaire d'un nom au lieu d'un numéro de ligne. Aussi sont-elles suffisamment explicites en elles-mêmes. Les appels de fonctions et de procédures prennent un sens à la seule lecture du programme. Ils ne sont pas modifiés par la renumérotation ou par la fusion du programme avec un autre programme. En outre ils sont exécutés bien plus rapidement que GOSUB et GOTO.

Les procédures et les fonctions commencent respectivement par DEF PROC et DEF FN. Elles comportent généralement une liste de paramètres. Par exemple (avec calc pour « calcul » et trait pour « traitement ») :

```
1200 DEF FNcalcta,b,c) = (a-b)*c/1000 et
2500 DEF PROCtrait(w,x$,y$,z)
```

Les définitions prennent ces paramètres pour des variables du programme. Cependant, lors de l'appel des fonctions ou des procédures par le programme, ces paramètres peuvent être remplacés par tout numéro de variable ou par toute expression littérale de même type de données que le paramètre originel.

Par exemple :

```
250 result = FNcalc (prix, coût, 12) ou
545 PROCtrait (6, noms$, « dupont », tableau (12))
```

Les valeurs des paramètres sont ensuite utilisées à la place des variables virtuelles dans la définition. Vous remarquerez qu'une fonction peut être mise dans une expression en tant que variable ou quantité arithmétique, alors qu'un appel de procédure doit être utilisé comme une commande BASIC. La commande LOCAL, qui définit des variables à n'utiliser que dans le bloc de définition, supprime le risque d'erreur sur le sous-programme. Soit le code :

```
100 FOR K = 1 TO 10:GOSUB 500:NEXT K:END
500 FOR K = 1 TO 5: PRINT« **** »:NEXT:RETURN
```

La variable K est ici utilisée comme compteur de boucles à la ligne 100 du programme principal, et à nouveau au sous-programme de la ligne 500. C'est là un enchaînement dangereux car trop éloigné, et qui ralentira l'exécution. Mais c'est un mal nécessaire dans la mesure où en BASIC classique comme ici, il sera très difficile à éviter, ou à examiner à la trace dans un long programme. Par contre, dans une procédure BASIC BBC, ce danger est contournable (avec aster pour « astérisque ») :

```
100 FOR K = 1 TO 10:PROCaster:NEXT END
500 DEF PROCaster
520 LOCAL K
540 FOR K = 1 TO 5:PRINT«****»:NEXT
560 ENDPROC
```

La commande LOCAL signifie qu'entre les lignes 500 et 560 la variable K est une nouvelle variable, indépendante de la variable K située ailleurs dans le programme, et n'ayant aucun effet sur la valeur de ce K ailleurs. Vous remarquerez que PROCstars est une procédure sans paramètres.

REPEAT...UNTIL est une forme syntaxique de boucle, dans laquelle l'itération se poursuit jusqu'à ce que l'expression conditionnelle suivant UNTIL soit *vraie*. Le contrôle passe alors sur l'instruction suivant UNTIL. Par exemple :

```
200 DATA 12,234,31,45,65,0,76,81
250 REPEAT
300 READ nombre:somme = somme + nombre
350 UNTIL nombre = 0
400 PRINT « la somme est de »; somme
```

ENTRE LES LIGNES

N'oubliez pas de numéroter vos lignes par multiples de 10. Même les meilleurs programmeurs se trouvent parfois obligés de faire des insertions entre les lignes...

Cette dernière syntaxe est beaucoup plus lisible et bien moins sujette à erreurs que les boucles GOTO ou FOR...NEXT virtuelles.

Le BASIC BBC comporte des aides au dépistage d'erreurs : TRACE, ON ERROR, et ERL. TRACE permet de suivre à la trace, et à l'écran, les lignes de programmes exécutées. ON ERROR GOTO ou ON ERROR GOSUB signifie qu'en cas d'erreur fatale à la bonne exécution du programme (y compris pour la touche d'« échappement », ESCAPE) le contrôle passera sur une routine de traitement des erreurs définie par l'utilisateur (ce peut être l'affichage de la valeur des variables). ERL est une variable système comportant le numéro de la ligne responsable de l'erreur.

Le micro BBC comporte de multiples extensions BASIC, comme les appels du système d'exploitation, la commande de l'unité de visualisation (le moniteur), les diverses variables système, ainsi que l'assembleur.

Le graphique BBC

Les commandes graphiques BASIC du micro BBC sont les suivantes :

MODE
Détermine le mode d'affichage par MODE N, où N = 0 à 7 :

Mode	Graphisme	Couleurs	Texte
0	640 × 256	2	80 × 32
1	320 × 256	4	40 × 32
2	160 × 256	16	20 × 32
3	Texte seul	2	80 × 25
4	320 × 256	2	40 × 32
5	160 × 256	4	20 × 32
6	Texte seul	2	40 × 25
7	Télétexte		40 × 25

Les micro-ordinateurs BBC modèle A n'ont accès qu'aux modes 4, 5, 6 et 7. Le modèle B permet l'accès à tous les modes graphiques. Pour les modes 0 à 6, il est possible de modifier le jeu de caractères par la commande VDU (de visualisation). Les caractères télétexte du mode 7 sont intangibles et ne correspondent pas au code standard ASCII.

COLOUR
Attribue une couleur parmi les seize au texte et au fond, selon le mode retenu.

COLOUR N
N est compris entre 0 et 15 pour la couleur du texte, et entre 128 et 143 pour la couleur du fond. La couleur attribuée par chaque valeur de N ne reste pas la même d'un mode à l'autre. Le guide de l'utilisateur donne la correspondance entre les valeurs de N et les couleurs, selon le mode.

VDU
Commande très utile, VDU A équivaut à PRINT CHR$(A). De même, VDU A,B,C a le même effet que PRINT CHR$(A);CHR$(B);CHR$(C); cela veut dire que toutes les routines de graphique et de texte sous le contrôle de codes 32CHR$, qui sont en doublet par rapport à la plupart des commandes graphiques BASIC, pourront être mises en œuvre avec quelques commandes VDU.

CLG
Vide la partie graphique de l'écran courant, et rapatrie le curseur sur sa position de départ, dans le coin inférieur gauche de l'écran.

CLS
Vide la partie texte de l'écran, et rapatrie le curseur sur sa position de départ, en haut à gauche de l'écran. Tout graphique sera également effacé.

DRAW
Tire des lignes à l'écran selon les modes 0, 1, 2, 4 et 5. De syntaxe :

DRAW X,Y

Le point défini par les coordonnées X et Y représente la fin de la ligne. Le point de départ peut être soit le point terminal de la dernière ligne tracée, soit un point défini par une commande MOVE.

GCOL
Détermine les couleurs courantes de fond et d'affichage. La syntaxe en est :

GCOL N,M

N donne l'emploi des couleurs (0 à 4); M définit la couleur logique selon les mêmes principes que COLOUR. N a les effets suivants :
— 0 : passe la couleur spécifiée par M;
— 1 : OU logique (OR) M avec la couleur présente;
— 2 : ET logique (AND) M avec la couleur présente;
— 3 : OU logique exclusif (OR) M avec la couleur présente;
— 4 : inverse la couleur présente.

MOVE
Positionne le curseur graphique sur un point déterminé par :

MOVE X,Y

Avec le même effet que DRAW, mais sans tirer de ligne.

PLOT
Utilisable pour de nombreuses fonctions graphiques, y compris le tracé d'un point, d'une ligne et d'un triangle :

PLOT K,X,Y

K définit le type de graphique tracé (par PLOT) selon des valeurs comprises entre 0 et 255. Cela détermine le type de lignes à tracer et les couleurs qu'elles prennent selon la liste figurant dans le guide de l'utilisateur.

POINT
Donne le numéro en rapport avec la couleur de l'écran logique, pour la coordonnée spécifiée :

NUMVAR = POINT (X,Y)

NUMVAR est une variable numérique.

Moniteur de mémoire

Le système de numération hexadécimale paraît bien compliqué, mais il se révèle un moyen extrêmement utile pour comprendre le maniement des adresses et de leur contenu.

A ce stade de notre cours de code machine, revenons en arrière pour revoir la question de la représentation des nombres. Le système décimal, que nous utilisons la plupart du temps, nous est déjà familier, et nous avons étudié le système binaire. Il ne faut pas oublier que ces deux systèmes ne sont que des expressions différentes du même concept : le nombre. La plupart des humains, par exemple, ont le même nombre de doigts à chaque main. On peut dire qu'il y en a cinq, d'autres diront *five, fünf* ou *penta*; mais nous voyons tout de suite qu'il s'agit de la même quantité ou du même nombre — seul le système de représentation diffère. Il est différent mais équivalent. Il y a correspondance univoque entre tous les nombres, qu'ils soient exprimés en français ou dans une autre langue, et tous ces systèmes possèdent une logique interne. L'arithmétique conduit aux mêmes résultats, quelle que soit la langue utilisée pour décrire les termes d'une expression arithmétique.

Les systèmes de numération sont tout à fait analogues à différentes langues. Le nombre de doigts d'une main humaine normale ne varie pas suivant qu'on l'appelle cinq ou *five,* ni si on l'écrit 5 ou 101b (b désignant ici le système binaire). Les seules raisons de choisir l'un ou l'autre système sont soit l'usage, soit la commodité.

Il nous semble pratique d'utiliser la représentation décimale, d'abord parce que c'est le système le plus communément employé autour de nous. Mais ce n'est pas le seul. Les montres numériques, par exemple, font appel à un curieux système d'arithmétique : en partie décimal, en partie modulo 60 (il y a 60 minutes dans 1 heure et 60 secondes dans 1 minute) et en partie modulo 24 (24 heures par jour). Avant 1971, la monnaie britannique se comptait par unités de 12 (pence dans un shilling) et de 20 (shillings dans une livre). Il fallait des années pour apprendre ce système aux écoliers — et combien de gens savaient vraiment comment exprimer des shillings et des pence en fractions décimales d'une livre?

En parlant d'ordinateurs, il nous a semblé instructif de commencer par les nombres binaires, car ils représentent très bien le fonctionnement électrique des ordinateurs, qui n'est autre qu'une suite d'états de commutateurs ouverts ou fermés. Si nous ne voulions parler que d'octets, le système binaire remplacerait avantageusement le système décimal — il est extraordinairement facile de transcrire un octet binaire en nombre décimal avec un peu de pratique. Malheureusement, les adresses de mémoire en particulier et les nombres utiles en général sont habituellement trop grands pour s'écrire dans un octet, de sorte que les programmeurs et les ingénieurs informaticiens ont ressenti la nécessité d'utiliser un système de numération compatible avec le système binaire, tout en ayant la portée du décimal. Deux systèmes ont ainsi été choisis : l'hexadécimal et l'octal. Le premier, devenu standard en micro-informatique, est abrégé en *hex* et fondé sur le nombre 16. L'octal, basé sur 8, a également été largement appliqué, mais il est progressivement remplacé par le système hex.

L'utilisation des nombres hex

En considérant les représentations décimale et binaire, nous avons vu deux conséquences du choix de la base numérique : la base est le nombre de chiffres nécessaires dans le système, et c'est le facteur multiplicatif dans la notation positionnelle. Par exemple, il y a dix chiffres dans le système décimal (0 à 9), et la valeur d'un chiffre décimal est multipliée par dix chaque fois qu'il se décale vers la gauche dans un nombre décimal.

L'hexadécimal requiert donc 16 chiffres : ce sont les chiffres 0 à 9 et les lettres A à F. Compter en hex ne consiste qu'à manipuler les chiffres et les réutiliser en notation positionnelle. Le nombre hex suivant 9 est donc A (10 décimal); ensuite vient B; puis C; et ainsi de suite jusqu'à F (15 décimal). On a épuisé les chiffres, et le nombre suivant F est donc 10 (prononcez « un zéro hex ») qui correspond au 16 décimal. Cela nous montre comment utiliser les chiffres, la valeur des colonnes dans un nombre hex à plusieurs chiffres étant multipliée par 16 à chaque décalage vers la gauche. Dans les nombres décimaux, nous appelons les colonnes : unités, dizaines, centaines et milliers. En comparaison, dans un nombre hex les colonnes sont : unités, seize, deux cent cinquante-six et quatre mille quatre-vingt-seize. Par rapport au changement de colonnes en binaire, on voit l'avantage majeur des nombres hex : au lieu d'un nombre binaire à quatre bits, il suffit d'un seul chiffre hex (0 à 15 décimal). Voici quelques exemples pour illustrer cela.

Décimal	Binaire	Hex
0	00000000	0
1	00000001	1
2	00000010	2
3	00000011	3
.....		
7	00000111	7
8	00001000	8
9	00001001	9
10	00001010	A
11	00001011	B
12	00001100	C
13	00001101	D
14	00001110	E
15	00001111	F
16	00010000	10
17	00010001	11
.....		
24	00011000	18
25	00011001	19
26	00011010	1A
27	00011011	1B
.....		
31	00011111	1F
32	00100000	20
33	00100001	21

Un octet de 8 bits s'écrit donc avec huit chiffres binaires ou deux chiffres hex :

0 à 255	en décimal
00000000 à 11111111	en binaire
0 à FF	en hex

Pour convertir un nombre hex en binaire, il suffit donc d'exprimer chaque chiffre hex comme un nombre binaire de 4 bits. Si un nombre d'un octet s'écrit comme un nombre hex de deux chiffres, alors le chiffre hex de gauche correspond aux quatre chiffres binaires de gauche, tandis que le chiffre hex de droite correspond aux quatre chiffres binaires de droite. En coupant ainsi un octet en deux, on obtient deux demi-octets. Le demi-octet de gauche, correspondant au chiffre hex de gauche, est appelé demi-octet supérieur ou plus significatif; et celui de droite, l'inférieur ou le moins significatif. Voici un exemple :

```
                Demi-octet  Demi-octet
                supérieur   inférieur
                    ↓           ↓
206 =            1100        1110      = C E
 ↑                  ↑                    ↑
décimal    équivalent binaire        équivalent hex
```

Il est important de se familiariser autant que possible avec le système hexadécimal, pour la simple raison qu'il est plus facile de manipuler les nombres hex que les octets binaires. Pour vous en convaincre, il vous faudra un peu de pratique non seulement avec des exemples numériques, mais surtout avec des adresses de mémoire et les contenus des octets de mémoire. Une fois que vous aurez compris cela — et ce sera très bientôt —, vous vous demanderez comment vous avez pu employer le système décimal.

Dans ce numéro du cours de code machine, nous donnons des programmes pour le BBC Micro, le Commodore 64 et le Spectrum, qui nous permettent de regarder le contenu de certains octets de mémoire. Ces programmes, que nous avons appelés « Mempeek », vous demandent d'abord de spécifier l'« adresse de départ », puis de donner le nombre d'octets que vous voulez considérer. Si, par exemple, vous spécifiez l'octet 1953 comme point de départ et si vous souhaitez que le contenu des quatre octets suivants soit affiché, alors l'écran montrera le nombre décimal 1953 dans la colonne de gauche, puis listera les contenus des octets 1953, 1954, 1955 et 1956 dans les quatre colonnes suivantes.

N'oubliez pas que, si la machine montre que l'octet 1956 contient le nombre décimal 175, cela signifie que, dans l'une des puces de mémoire, une zone que la machine appelle l'octet 1956 comprend une séquence de huit niveaux de tension. Si 0 V est représenté par 0, et 5 V par 1, alors l'octet 1956 porte une séquence de tension de 10101111. Nous choisissons d'interpréter cela comme un nombre binaire, dont l'équivalent décimal est 175.

Il est essentiel de se rappeler que nous utilisons une sorte de raccourci presque chaque fois que nous parlons d'ordinateurs; traduire cela en termes physiques est toujours salutaire et devrait aider à éviter des confusions.

Les contenus d'un octet affichés sur l'écran ne sont pas les contenus « réels ». Ce que nous voyons, ce sont des caractères qui ont été assignés aux séquences de tension des octets. En interprétant ces valeurs comme des nombres binaires, et en convertissant ces derniers en décimaux, nous franchissons une étape de plus pour convertir les nombres décimaux en caractères ASCII *(American Standard Code for Information Interchange)*. C'est ce caractère qui est affiché sur la dernière colonne. Ce code, internationalement reconnu et appliqué à la plupart des ordinateurs, substitue des nombres décimaux compris entre 0 et 127 à tous les caractères d'un clavier (historiquement, un clavier de téléscripteur). Dans ce code, le nombre décimal 65 correspond au caractère majuscule A, 66 au B, 67 au C, et ainsi de suite. Parmi les caractères non alphabétiques, 32 représente l'espace, 42 l'astérisque, 13 le retour de chariot, etc.

Les caractères ASCII pouvant être imprimés commencent au 32 et se terminent au 127. En dehors de cet intervalle, les codes sont soit indéfinis, soit non imprimables, soit spécifiques à des machines particulières. C'est pourquoi, lorsque nous faisons tourner les programmes Mempeek, le moniteur affiche un point pour tout octet contenant un tel nombre. Dans le prochain numéro, nous verrons l'ensemble des caractères ASCII compris entre 0 et 127.

Cette investigation est particulièrement utile pour comprendre le code machine, pour deux raisons importantes. Premièrement, cela met en évidence le fait que l'interprétation des contenus de mémoire est entièrement une question de

choix. On peut dire qu'une adresse contient un nombre, une adresse, un caractère codé, une instruction, ou tout ce que vous voudrez. Deuxièmement, cela donne une notion bien plus claire de la mémoire, en particulier des zones de mémoire qui contiennent des données, dont certaines sont utilisées par le système d'exploitation de la machine et d'autres par vous.

Les données du système d'exploitation comprennent tous les messages d'erreur ou d'attente — tels que READY, NONSENSE IN BASIC, etc. —; tout ce qui peut être dit à l'utilisateur est codé en ASCII et stocké en mémoire. Vous n'y avez peut-être jamais pensé, mais ce coup d'œil à l'intérieur de la machine révèle les limites de son « intelligence ». Il est évident que notre intelligence est d'une autre nature : nous ne mémorisons pas de messages ainsi, nous concevons une pensée, puis formons une combinaison appropriée de mots pour l'exprimer.

Tables d'implantation en mémoire

Une table d'implantation en mémoire est une simple représentation schématique de l'usage auquel est destiné la mémoire, en montrant les paramètres des zones spécifiques. Certaines zones de mémoire ont toujours la même destination. Sur le Commodore 64, par exemple, les octets 0 à 1024 sont utilisés par le système d'exploitation BASIC comme zone de travail. D'autres zones ont divers usages selon la taille et l'état du programme. Les limites entre ces zones peuvent être soit *fixes* (traits pleins des schémas ci-dessous), soit *flottantes* (pointillés). Les limites fixes ne varient jamais, tandis que les limites flottantes délimitent des zones de mémoire qui fluctuent selon les besoins. Sur le Commodore, les limites de RAM d'écran sont fixes (octets 1024 à 2048), tandis que celles de la mémoire où sont stockées les variables BASIC fluctuent selon le nombre de variables utilisées.

Les programmes Mempeek de la page suivante peuvent servir à localiser les positions en cours des limites flottantes de mémoire dans votre machine. Le Commode a six pointeurs de limite flottante (également appelés variables système). Un exemple ci-dessous explique comment les contenus d'une paire d'octets sont utilisés pour calculer l'adresse de mémoire demandée. Le BASIC BBC comprend quatre variables système, et le Spectrum cinq.

Rappelons qu'une table d'implantation en mémoire est une représentation statique de quelque chose qui change constamment pendant que la machine fonctionne. Chacune des limites flottantes est susceptible de changer n'importe quand. A la page suivante, nous vous donnons des idées pour étendre les programmes Mempeek afin d'observer les variations des variables de pointeurs.

Variables système du Commodore

43,44 Début du TEXTE de PROGRAMME BASIC.
45,46 Début des VARIABLES BASIC.
47,48 Début des TABLEAUX BASIC.
49,50 Fin des TABLEAUX BASIC.
51,52 Bas du STOCKAGE de CHAINES BASIC.
55,56 Haut du STOCKAGE de CHAINES BASIC.

Exemple
Utilisez le programme Mempeek de la page suivante pour inspecter les contenus de ces octets. Votre écran pourrait, par exemple, afficher les nombres suivants :

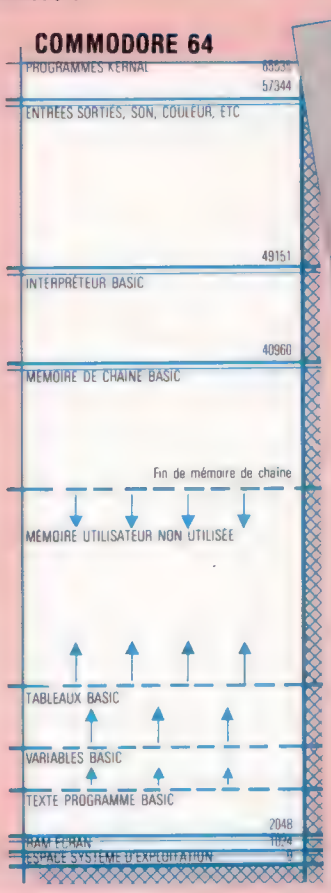
43 0 8 11 9

La première colonne est l'adresse du premier octet atteint. Les deuxième et troisième colonnes affichent le contenu des octets 43 et 44. Ce sont les octets de décalé et de page de l'adresse du début de la zone de texte BASIC.

8*256 + 0 = 2048

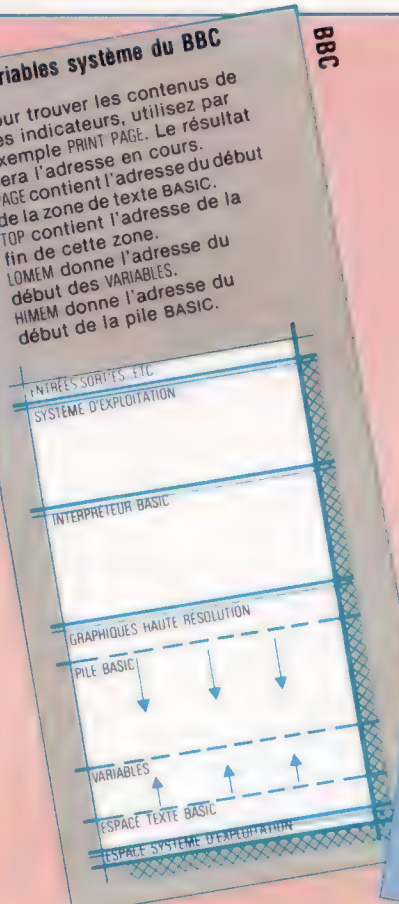
Les quatrième et cinquième colonnes sont les octets de décalé et de page pour la fin de la zone de texte BASIC. L'adresse est ainsi calculée :

9*256 + 11 = 2315



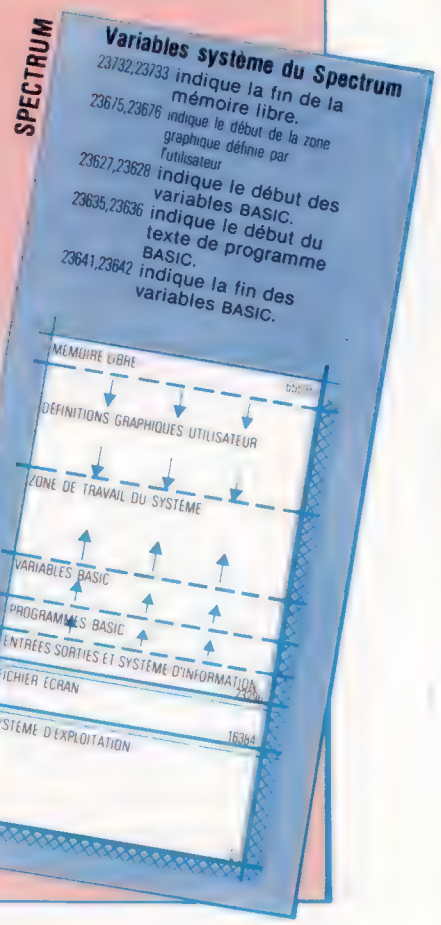
Variables système du BBC

Pour trouver les contenus de ces indicateurs, utilisez par exemple PRINT PAGE. Le résultat sera l'adresse en cours.
PAGE contient l'adresse du début de la zone de texte BASIC.
TOP contient l'adresse de la fin de cette zone.
LOMEM donne l'adresse du début des VARIABLES.
HIMEM donne l'adresse du début de la pile BASIC.



Variables système du Spectrum

23732,23733 indique la fin de la mémoire libre.
23675,23676 indique le début de la zone graphique définie par l'utilisateur
23627,23628 indique le début des variables BASIC.
23635,23636 indique le début du texte de programme BASIC.
23641,23642 indique la fin des variables BASIC.



BBC Micro

```
7    REM ******************************
8    REM *      BBC       MEMPEEK 1 *
9    REM ******************************
20   MODE 7
30   *TV 255
40   CLS
50   REPEAT
100  INPUT "ADRESSE DE DEPART",AD
200  INPUT "NOMBRE D'OCTETS (0 POUR ARRETER)"
     ,NO
250  PRINT "**************************************"
300  FOR B% = AD TO (AD + NO - 1) STEP 4
350  H$ = "":  X = 6
400  PRINT TAB(0);B$;TAB(8);
450     X = 4
500  FOR C = 0 TO 3
550  PK% = ?(B% + C):PK$ = "."
600  PRINT PK%;
650  IF PK% = 13 THEN PK$ = CHR$(124)
700  IF (PK% > 31) AND(PK%<128) THEN PK$ = CHR$(PK%)
750  H$ = H$ + PK$
800  NEXT C
850  PRINT TAB(32);H$
900  NEXT B%
950  UNTIL NO = 0
1000 REM ******************************
```

Spectrum

```
7    REM ******************************
8    REM *      SPECTRUM      MEMPEEK 1 *
9    REM ******************************
30   DIM H$(4)
50   FOR L = 0 TO 1 STEP 0
100  INPUT "ADRESSE DE DEPART";AD
200  INPUT "NOMBRE D'OCTETS (0 POUR
     ARRETER) " ;NO
250  PRINT "**************************************"
300  FOR B = AD TO (AD + NO - 1) STEP 4
350  LET H$ = "...."
400  PRINT B;TAB 7;
500  FOR C = 0 TO 3
550  LET PK = PEEK (B + C)
600  PRINT PK;" ";
650  IF (PK > 31) AND (PK < 128) THEN LET
     H$(C + 1) = CHR$ PK
700  IF PK = 13 THEN LET H$(C + 1) = " "
800  NEXT C
850  PRINT TAB 26;H$
900  NEXT B
950  IF NO = 0 THEN LET L = 2
1000 NEXT L
1050 REM ******************************
```

Commodore 64

```
7    REM ************************************
8    REM *      COMMODORE     MEMPEEK 1    *
9    REM ************************************
30   PRINT CHR$(147)       :REM EFFACE L'ECRAN
40   PRINT CHR$(142)       :REM MAJUSCULES
50   FOR LP = 0 TO 1 STEP 0
100  INPUT "ADRESSE DE DEPART";AD
200  INPUT "NOMBRE D'OCTETS (0 POUR ARRETER) "
     ;NO
250  PRINT "**************************************"
300  FOR B = AD TO (AD + NO - 1) STEP 4
350  H$ = ""
400  PRINT B;TAB(8);
500  FOR C = 0 TO 3
550  PK = PEEK(B + C):PK$ = "."
600  PRINT TAB(8 + 5 * C);PK;
650  IF PK = 0 THEN PK$ = CHR$(122)
700  IF (PK > 31) AND (PK < 128) THEN PK$ = CHR$(PK)
750  H$ = H$ + PK$
800  NEXT C
850  PRINT TAB(32);H$
900  NEXT B
950  IF NO = 0 THEN LP = 1
1000 NEXT LP
1050 REM ************************************
```

Utilisation de Mempeek

En entrant le programme Mempeek dans votre machine, assurez-vous que vous le sauvegardez et vérifiez-le soigneusement avant de faire RUN, parce que les erreurs de frappe dans ce type de programme peuvent conduire à des catastrophes irrémédiables. Le programme vous demandera d'abord une adresse de départ, puis le nombre d'octets que vous voulez examiner. Tous deux doivent être des nombres entiers positifs compris entre 0 et 65535. Si vous entrez 0 comme nombre d'octets, vous arrêterez le programme. Supposons que vous vouliez que l'adresse de départ soit l'octet 230. L'écran affichera ceci :

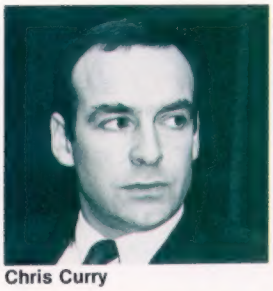
```
ADRESSE DE DEPART ? 230
NOMBRE D'OCTETS (0 POUR ARRÊTER) ? 8
**************************************

230   193   32   65   49   .A1
234   129   64   93   98   .@]b
ADRESSE DE DEPART?
```

La colonne de gauche donne l'adresse décimale du premier octet, les quatre colonnes suivantes donnent le contenu décimal des quatre octets à partir de cette adresse, et la dernière colonne donne la représentation des caractères contenus dans les octets (lorsque c'est possible) et «.» sinon.
Vous pourrez commencer par « vagabonder » dans la mémoire avec ce programme, en notant les adresses intéressantes, puis essayer de trouver à quel endroit de la mémoire le système d'exploitation stocke ses messages d'erreur et mots clés BASIC. Votre manuel d'utilisateur pourra vous être utile en cela.
Une fois que vous aurez trouvé les pointeurs qui délimitent les différentes zones de mémoire, vous pourrez essayer d'ajouter quelques lignes REM au programme, et voir leur effet sur les valeurs des pointeurs. Ajoutez ensuite quelques lignes au début du programme pour faire des manipulations de chaînes, puis regardez à nouveau l'effet produit sur les pointeurs et sur les contenus de la zone de stockage de variables.
Par exemple :

```
3 DIM Z$(254)
4 LET X$ = «»
5 FOR M = 1 TO 255: LET X$ = X$ + «"»: NEXT M
```

Petit à petit...

Acorn Computers a produit deux des plus beaux exemples de la micro-informatique britannique : le BBC Micro et l'Electron. Il y a quelques années à peine, cette firme naissait.



Chris Curry

Herman Hauser

Acorn

Le fondateur d'Acorn, Chris Curry, est un ancien employé — et un ami très proche — de sir Clive Sinclair, qui l'engagea en 1965 pour le salaire royal de 11 livres par semaine (environ 150 francs!).

Devenu employé de la Sinclair Radionics, Curry, ingénieur de recherche, fut nommé responsable d'un projet qui aboutit à la mise sur le marché de la calculatrice Executive, en 1971. Au cours des cinq années qui suivirent, il se consacra à la mise au point de machines de ce type, où l'on voit aujourd'hui les ancêtres directs des micro-ordinateurs domestiques. En 1975, Sinclair Radionics cessa toute activité, et Curry s'associa avec Sinclair pour une nouvelle opération nommée « Science in Cambridge » : il s'agissait de vendre des composants électroniques sous forme de kits.

Une calculatrice montre-bracelet se vendit particulièrement bien. Mais Curry s'intéressait aux ordinateurs construits à partir d'un circuit imprimé, tels qu'ils commençaient à apparaître aux États-Unis. Il entreprit d'en construire un. Ce fut le MK 14 (microprocesseur à 14 puces, livré en kit), qui comprenait un microprocesseur de chez National Semiconductors, une RAM de 256 octets, une petite ROM abritant un programme moniteur, et les composants nécessaires pour un affichage à huit chiffres, utilisant des diodes électroluminescentes.

Acorn

Curry s'aperçut aussi que la compagnie était sans cesse amenée à fournir conseils et idées aux amateurs d'électronique qui lui téléphonaient; il décida d'engager Herman Hauser, qui préparait alors un doctorat à Cambridge, afin de s'occuper de ces problèmes. Mais il eut bientôt de profondes divergences avec Sinclair, et décida de fonder sa propre compagnie. Ce fut la Cambridge Processor Unit (en abrégé CPU; allusion à *central processing unit,* « unité centrale »). Hauser devint son associé, et tous deux, installés dans un petit bureau de Bridge Street, à Cambridge, se firent experts-conseils en électronique et en informatique.

Le succès du MK 14 et l'évolution du marché aux États-Unis montraient clairement que les amateurs étaient à la recherche d'un ordinateur intégré dans un boîtier, et qui parlerait BASIC. Au cours d'un projet pour un client, CPU avait rédigé une version très rapide de ce langage, destinée au contrôle des machines; la firme décida de l'implanter sur un appareil qu'elle mettrait en vente. Ce fut l'Atom, et à cette occasion CPU prit le nom d'Acorn. Il s'agissait avant tout de conquérir le marché éducatif; mais beaucoup de lycées estimèrent que le BASIC utilisé différait par trop des standards établis par Microsoft pour pouvoir être acceptable. Toutefois l'Atom eut beaucoup de succès auprès des particuliers. Une version améliorée, le Proton, fut mise au point à l'intention des laboratoires et des universités.

Le Proton en était au stade de préproduction lorsque, en 1981, Curry apprit que la B.B.C. était à la recherche d'un appareil permettant la mise en application de ses cours d'informatique. Il devait être d'usage facile, même pour les débutants, avoir de grosses possibilités, et pourtant rester de prix raisonnable (le prix imposé était de 200 livres, soit environ 2 500 francs).

Curry entreprit non d'améliorer encore le Proton, mais de construire quelque chose de nouveau, qui ferait la démonstration des capacités du microprocesseur 6502. Après une lutte sans merci face à Sinclair Research (qui proposait le Spectrum), Acorn obtint le contrat, et sortit le micro-ordinateur BBC.

PROGRAMME N° 14

LES INSTRUCTIONS READ ET DATA

L'instruction DATA permet de prédéfinir des données à l'intérieur du programme lui-même. Ces données sont ensuite lues dans les variables choisies par l'instruction :

« READ variable »

Ces DATAS sont en général données en tout début de programme. L'ordre dans lequel sont saisies les données détermine l'indice de ces mêmes données.

Exemple :

```
$LIST

5  REM SAISIE DES DATAS
10 DATA 16, 15, 0, 3
15 REM LECTURE DES DATAS
20 READ A1
30 READ A2
40 READ A3
50 READ A4
```

Dans ce programme, la machine lit successivement les valeurs 16, 15, 0, 3 dans les variables :

A1, A2, A3, A4

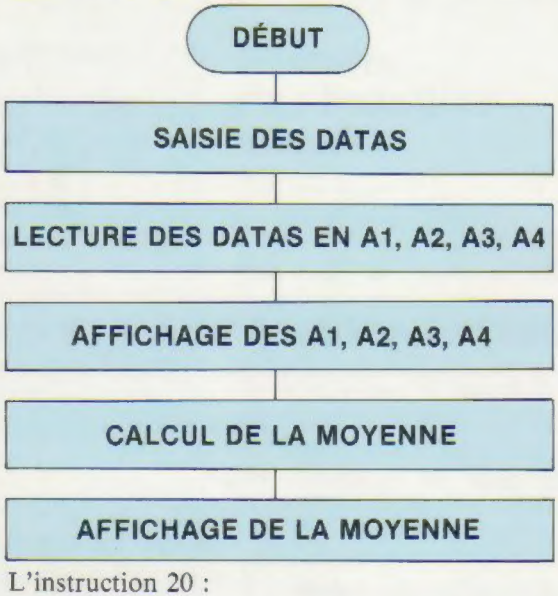
A l'issue de quoi : A1 = 16
A2 = 15
A3 = 0
A4 = 3

A ce début de programme ajoutons, par exemple, un petit calcul de moyenne. On obtient :

```
$LIST

5   REM SAISIE DES DATAS
10  DATA 16, 15, 0, 3
15  REM LECTURE DES DATAS
20  READ A1
30  READ A2
40  READ A3
50  READ A4
52  REM CALCUL DE MOYENNE
55  PRINT "CALCUL DE MOYENNE": PRINT : PRINT
57  REM AFFICHAGE DES NOTES
60  PRINT "PREMIERE NOTE :";A1
70  PRINT "DEUXIEME NOTE :";A2
80  PRINT "TROISIEME NOTE :";A3
90  PRINT "QUATRIEME NOTE :";A4
95  REM ON SAUTE DES LIGNES
100 PRINT : PRINT : PRINT
107 REM ON IMPRIME LA MOYENNE
110 INVERSE : PRINT "MOYENNE :";(A1 + A2 + A3 + A4) / 4
120 NORMAL
130 END
```

```
$RUN
CALCUL DE MOYENNE

PREMIERE NOTE : 16
DEUXIEME NOTE : 15
TROISIEME NOTE : 0
QUATRIEME NOTE : 3

MOYENNE : 8.5
```

dont l'organigramme est :



L'instruction 20 :

```
20 READ A1
```

lit la première donnée 16 dans A1. Le compteur de DATA (géré par le BASIC) progresse de 1.
L'instruction 30 :

```
30 READ A2
```

lit la deuxième donnée dans A2, etc.
On peut utiliser l'instruction READ de manière différente en indiquant, par exemple, la fin des DATAS par la note 21. On obtient le programme suivant :

```
$LIST

4    REM ON MET LE CONPTEUR TT A 0
5    TT = 0
7    REM SAISIE DES DATAS
10   DATA 16, 15, 0, 3, 21
15   REM LECTURE DES DATAS
20   READ ZN
25   REM QUAND ZN = 21 IMPRIMER LA
     MOYENNE
30   IF ZN = 21 THEN GOTO 1000
45   PRINT ZN
50   TT = TT + ZN
60   NT = NT + 1
70   GOTO 20
100  REM ON IMPRIME LA MOYENNE
1000 PRINT "MOYENNE DES NOTES :";
     TT / NT

$RUN
16
15
0
3
MOYENNE DES NOTES : 8.5
```

Grâce à la variable TT, on dispose d'un compteur que l'on initialise à 0 en début de programme et auquel on rajoute la valeur de chaque ZN après chaque impression (ligne 50). De même, en NT on dispose d'un compteur de note qui permet ainsi d'obtenir directement la moyenne en 1 000. Moyenne, donc, qui vaut TT/NT (TT, total de toutes les notes; NT, nombre de notes).
DATA peut être utilisé aussi sur plusieurs lignes. Par exemple :

```
10 DATA 100, 200, 0, 50
20 DATA  50, 25
30 DATA  20
```

Ce qui est équivalent à la ligne :

```
10 DATA 100, 200, 0, 50, 50, 25, 20
```

DATAS ET CARACTÈRES NON NUMÉRIQUES

Les DATAS contenant des caractères alphanumériques doivent se placer entre guillemets. Si les guillemets sont absents, la virgule fera office de séparateur. Voici un petit exemple :

```
$LIST

5  REM SAISIE DU DATA ALPHANUMERIQUE
10 DATA "45 RUE DE LA PAIX"
15 REM LECTURE DU DATA
20 READ Z$
25 REM ON IMPRIME LE DATA
40 PRINT Z$

$RUN
45 RUE DE LA PAIX
```

On peut aussi considérer des nombres comme caractères alphanumériques et donc être lus comme une chaîne de caractères. Ces nombres doivent être placés entre guillemets. En voici un petit exemple :

```
$LIST
5  REM SAISIE DU DATA
10 DATA "500000"
15 REM LECTURE DU DATA
20 READ Z$
25 REM ON IMPRIME LE DATA
30 PRINT Z$
35 REM ON IMPRIME LES 2 CARACTERES DE
   GAUCHE DU DATA
40 PRINT LEFT$ (Z$, 2)

$RUN
500000
50
```